SOCR *Motion Charts*: An Efficient, Open-Source, Interactive and Dynamic Applet for Visualizing Longitudinal Multivariate Data

Jameel Al-Aziz
Nicolas Christou
Ivo D. Dinov
University of California, Los Angeles

Journal of Statistics Education Volume 18, Number 3 (2010),
www.amstat.org/publications/jse/v18n3/dinov.pdf

Key Words: Statistics education; Exploratory data analysis; Motion-charts; SOCR; Java; Applets; Activity.

Abstract

The amount, complexity and provenance of data have dramatically increased in the past five years. Visualization of observed and simulated data is a critical component of any social, environmental, biomedical or scientific quest. Dynamic, exploratory and interactive visualization of multivariate data, without preprocessing by dimensionality reduction, remains a nearly insurmountable challenge. The Statistics Online Computational Resource (www.SOCR.ucla.edu) provides portable online aids for probability and statistics education, technology-based instruction and statistical computing. We have developed a new Java-based infrastructure, SOCR *Motion Charts*, for discovery-based exploratory analysis of multivariate data. This interactive data visualization tool enables the visualization of high-dimensional longitudinal data. SOCR *Motion Charts* allows mapping of ordinal, nominal and quantitative variables onto time, 2D axes, size, colors, glyphs and appearance characteristics, which facilitates the interactive display of multidimensional data. We validated this new visualization paradigm using several publicly available multivariate datasets including Ice-Thickness, Housing Prices, Consumer Price Index, and California Ozone Data. SOCR *Motion Charts* is designed using object-oriented programming, implemented as a Java Web-applet and is available to the entire community on the web at www.socr.ucla.edu/SOCR_*MotionCharts*. It can be used as an instructional tool for rendering and interrogating high-dimensional data in the classroom, as well as a research tool for exploratory data analysis.

1. Introduction / Background

The volume and complexity of data acquired, stored and processed by various organizations doubles every year (Safran et al., 2007). There are well-established protocols for data collection (Cohen et al., 2007). Yet, there are significant discipline, socio-economic and geo-political differences on how to manage, analyze and interpret experimental, observational or simulated data (Le Gal et al., 2006; Zimmer et al., 2006). Two primary limitations of data handling revolve around the access to meta-data (auxiliary descriptional data *explaining* the core data measurements), and the specific strategies for data presentation and interpretation. In this manuscript, we present a new efficient graphical approach for data representation and interrogation. Active data visualization is a critical component of any data understanding, as it provides visual, informative and quantitative cues to the data behavior and the intrinsic data characteristics. These features, in turn, drive our subsequent protocols for data modeling, quantitative analysis and rigorous interpretation. In addition, dynamic data visualization enables non-experts to navigate, explore and formulate hypotheses about the data, as well as identify data patterns, trends and challenges without high-level scientific and technical knowledge.

There are several other projects presenting web-based graphical data analysis tools. Examples include www.Swivel.com, www.Many-Eyes.com, www.Gapminder.org, and *Google* Motion Charts (Grossenbacher, 2008; Vermylen, 2008). These graphical resources allow users to interactively explore relationships and trends of data with temporal characteristics. The main differences between the SOCR *Motion Charts* and other similar interactive graphical data visualization environments are that the former is an open-source project that provides unrestricted anonymous Internet access to the applet, the source code, and the data, and it is implemented in a platform-agnostic Java framework.

In the Summer of 2009, we were asked to analyze a complex dataset that included observational multivariate ozone depletion data. The data included California Ozone measurements from 20 locations between 1980 and 2006 (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_121608_OzoneData). Figure 1 illustrates a snapshot of the dynamic interactive map of the geographic locations of the California ozone data measurements. The GoogleMaps-based interactive geographic map is available online and includes part of the information included in the Ozone dataset (e.g., longitude and latitude). Table 1 shows only 6 rows and 9 variables, while the complete dataset consists of 540 rows and 22 variables. The goals of the study were to identify relationships and associations between the variables and map geographically the significant ozone pollution effects. Any such quantitative study requires a preliminary exploratory data analysis. The complexity of the dataset and the intrinsic measurement characteristics of the ozone data demanded a new approach to visualization and exploration of these heterogeneous measurements. The complexity of the ozone dataset is rooted in the fact that it includes longitudinal (time), 3D spatial (longitude, latitude, altitude), seasonal (month), and meta-data characteristics (e.g., coverage rate). This data heterogeneity presents a problem with some rudimentary data analysis methods and is a motivation for the development of SOCR Motion Charts.

Figure 1: *Geographic Map of the California Ozone Layer Data.*

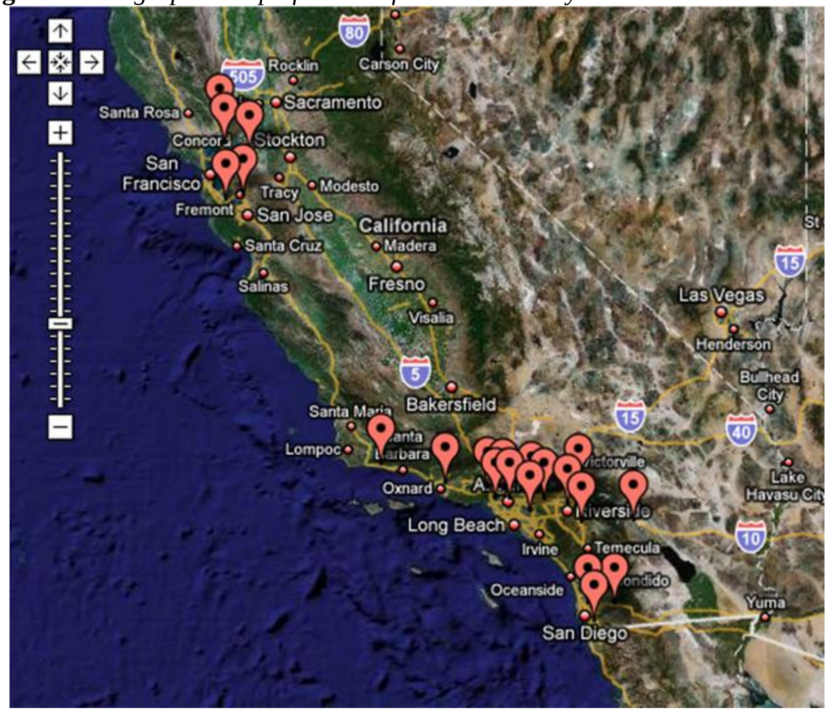


Table 1: *A fragment of the multivariate ozone layer data.*

MULTIVARIATE CALIFORNIA OZONE DATA									
Index	**Variable**	**Location**	**Year**	**Annual**	**Hi Cover**	**Comp Sites**	**Latitude**	**Longitude**	**Elevation**
1	OZMAX1HR	2008	1980	0.12	97	1	34.46222	-120.026	24
2	OZMAX1HR	2008	1981	0.11	95	1	34.46222	-120.026	24
3	OZMAX1HR	2008	1982	0.15	99	1	34.46222	-120.026	24
4	OZMAX1HR	2008	1983	0.14	98	1	34.46222	-120.026	24
5	OZMAX1HR	2008	1984	0.14	97	1	34.46222	-120.026	24
6	OZMAX1HR	2008	1985	0.13	99	1	34.46222	-120.026	24
…	…	…	…	…	…	…	…	…	…

The UCLA Statistics Online Computational Resource (www.SOCR.ucla.edu) is an NSF-funded project that provides integrated resources for probability, statistics and science education (Dinov, 2006; Dinov and Christou, 2009; Dinov et al., 2009). The main SOCR goals are:

- to provide a web-based open-access and open-development infrastructure for platform-independent probability and statistics tools;
- to maintain a well-designed, extensible and open-source computational library;
- to design efficient graphical user interfaces (GUI's) for various computational resources; and
- to support an integrated framework of course-materials, simulations, computations, and instructional resources.

The interactive SOCR Hyperbolic Graphical navigator provides easy navigation and discovery of SOCR learning, computational and instructional resources (www.socr.ucla.edu/SOCR_HT_ResourceViewer.html). A recent (Institutional Review Board approved) experimental study conducted with UCLA undergraduates students, showed significantly higher performance on quantitative tests (quizzes and examinations) of a SOCR treatment group versus a control group where traditional teaching methods were employed (Dinov et al., 2008). Another benefit of SOCR learning intervention was that student performance in the SOCR-treatment groups was more homogeneous – there was a reduction of the variability of test scores on quantitative examinations within each section of the treatment groups compared to control groups. The SOCR-treatment groups reported more satisfaction and found the courses more interesting compared to matched traditionally taught control groups (Dinov et al., 2008).

2. Motivation

In the past year, we had collected, explored and analyzed a variety of datasets of complexity similar to that of the California Ozone dataset. For instance, we studied large datasets of Housing Price Index (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_010309_HPI), California Earthquakes (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_021708_Earthquakes), Consumer Price Index (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_021808_CPI), and Polar Ice Thickness (http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_042108_Antarctic_IceThicknessMawson). All of these data shared several interesting characteristics – they had a temporal dimension, they were acquired longitudinally, and included multivariate measurements. The complexities of these data present a challenge in terms of their interactive visualization. For example, classical approaches for collective visualization of the entire earthquake dataset is difficult because of the multivariate and heterogeneous features of this dataset. To address these 3 challenges we designed, implemented and validated a new efficient graphical visualization tool for dynamic rendering of longitudinal multivariate data. The new tool, called SOCR *Motion Charts*, is a Java applet that enables mapping of quantitative and categorical variables into a variety of graphical widgets (e.g., glyphs, colors, 2D axes, sizes, etc.), which facilitates the interactive data exploration and discovery of intrinsic relations and patterns in the dataset.

SOCR *Motion Charts* aim to provide a new data visualization paradigm that facilitates the representation and understanding of large (volume) and multivariate (complex) data by non-expert users (e.g., students and learners). Many classical data visualization techniques have limitations in terms of the volume, properties or complexity of the dataset – e.g., regression plots require bivariate data, MS Excel charts limit the number of columns and rows to 256 and 65,536, respectively. SOCR *Motion Charts* enables the display of large multivariate data with thousands of data points and allows for better visualization of the data by using additional dimensions, (e.g., time, the size of the blobs, and color) to show different facets of the data. Hence the dynamic nature of the plots allows better extraction of trends in the temporal dimension and visualization of more characteristics simultaneously. Although there are no core-design limitations of Motion Charts in terms of the data volume and complexity, the size of the dataset may exceed the memory allocated by the browser to the Java applet (this is a user configurable option within most browsers), and the current implementation of Motion Charts enables mapping up to 6 variables. As all longitudinal datasets have temporal dimension, SOCR *Motion Charts* provides an interactive way to map variables to different graphical widgets and interactively traverse (play the chart) in the time dimension. This data rendering over the Internet using browser-embedded applets enables the exploration of the data and allows users to quickly formulate and address questions about various intrinsic data characteristics. SOCR *Motion Charts* facilitate the transformation of quantitative and qualitative information contained in multivariate data into intuitive, relevant, and actionable knowledge.

The central object of interest in motion charts is a *blob*. A blob is a solid disc, or in general a 2D shape, which represents one entity from the dataset – this may be one or more rows in a spreadsheet, according to the user selected variable mapping. Blobs have 3 important characteristics – size, color and position. Users may specify variable mappings which are used to determine the appearance of the blobs at different time points. For example, in the Housing Price Index (HPI) dataset, see Housing Prices section below, blobs can represent States and their size, color and position characteristics may indicate HPI values, region, population size, unemployment, etc. The 2D x-y positions of blobs are determined from the values of the variables mapped by the user onto the X and Y axes. For instance, in the HPI data, the state-specific unemployment rate (UR) and population-size (Pop) may be used to localize the blob for each state. Of course, blob appearance (color, size) and blob-locations (x-y coordinates) vary with time, which is the core of the motion charts data exploration. This dynamic appearance of the data in the Motion Charts facilitates the visual inspection of associations, patterns and trends in multivariate datasets.

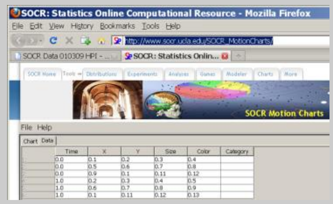
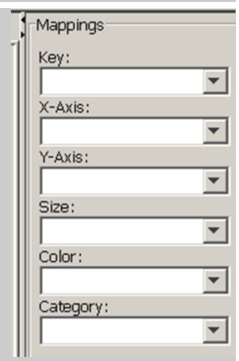
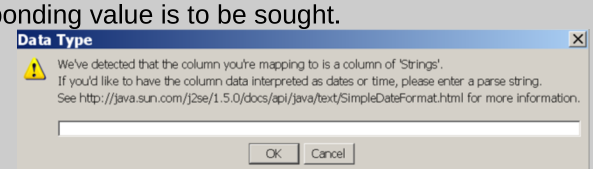
The variable mapping step is the most important part of the Motion Chart exploratory data analysis. In general, there is no *best* method to map data variables into Motion Charts appearance. However, some mappings may be more *appropriate* or *informative* under certain conditions. Both the data characteristics and the underlying investigative inquiry would drive the selection of a particular variable mapping. For instance, geographic, spatial or regional data may provide a natural mapping of location variables onto the X and Y axes. Quantitative and qualitative types of variables may be more appropriate for the size and category fields in the Motion Charts, respectfully. Varieties of ordinal variables can be mapped onto the different Motion Chart graphical widgets (e.g., glyphs, colors, 2D axes, sizes, etc.). Users may need to experiment with several variable mappings before they identify an appropriate mapping that illustrates interesting data characteristics relevant to the underlying driving research question(s), facilitating the interactive data exploration, and enabling discovery of the data interrelations, patterns or trends.

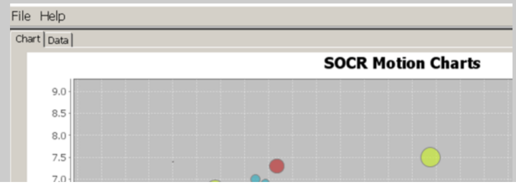
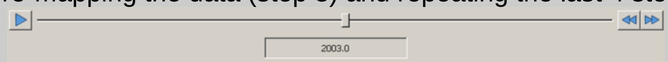
3. Motion Charts Usage

Table 2 illustrates the complete protocol for utilization of the SOCR *Motion Charts* as an exploratory data analysis tool. After going over this protocol, the readers are encouraged to repeat these steps using their own data. There are two screencast videos that demonstrate the utilization of the SOCR Motion Charts. These videos are very large (44MB) but should open quickly provided that you have a web browser open and a recent version of QuickTime installed. Videos follow:

	http://www.socr.ucla.edu/docs/videos/SOCR_CA_OzoneActivity_July2010.mp4 http://www.socr.ucla.edu/docs/videos/SOCR_HousingPriceActivity_July2010.mp4

Table 2: A step-by-step guide for utilization of SOCR *Motion Charts* using tabular multivariate data.

STEPS	DESCRIPTION
1. Go to www.socr.ucla.edu/SOCR_MotionCharts	Start the SOCR Motion Charts applet by pointing your Java-enabled web-browser to this URL address.  Note that the applet starts with a simple default dataset.
2. Load data in the applet	Use another browser tab, or open a new browser window, and identify appropriate data stored in a spreadsheet format (text, HTML, tabular, CSV, etc.) For example: http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_010309_HPI Use CTR+C or Apple+C to copy the data from your source document, click on the top-left most cell in the Data tab on the applet and use CTR+V or Apple+V, depending on your hardware platform to paste the data in the applet spreadsheet.
3. Data mapping	Next, map your variables (columns in your data table) to Motion Charts glyphs. This process allows control over the appearance, rendering and dynamics of the motion chart. For example, this allows you to specify the temporal variable (time) which will be used to animate the chart over.  Note: When mapping string (non-ordinal categorical) variables, the data do not have an implicit order. In these situations a dialog pop-up prompts the user to provide a hash-map for "ordering" these data. Click "OK" to go with a default hash-map based on *lexicographical* string-ordering. A hash map is a data structure that explicitly identifies values, known as keys, (e.g., string value) to their associated quantitative counterparts (e.g., order of the string). Similarly, hash mapping is used to transform the key into the index (the hash) of an array element (the slot) where the corresponding value is to be sought. 

4. Motion Chart rendering and visualization	Finally, go to the Chart tab on the Motion Chart applet. This will show you the motion chart at the initial time. 
5. Controls	Use the Play, Pause, Forward and Backward buttons to control the Motion of the Chart animation. You may want to experiment with re-mapping the data (step 3) and repeating the last 4 steps. 

4. Results

We present several complex examples illustrating the utilization of the SOCR *Motion Charts* to dynamically visualize time-varying multivariate data. The static images included in these examples have a limited power to showcase the real temporal changes. To get a better understanding of these examples, the reader is encouraged to copy-and-paste the referenced data into the SOCR *Motion Charts* applet and interact hands-on with the applet and the datasets provided here or copy and paste other multivariate data into the applet.

4.1 Housing Prices

(http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_010309_HPI)

The housing price data were provided by the Office of Federal Housing Enterprise Oversight and are shown in Table 3. The data represent the average housing price for all states between the years 2000 and 2006. The complete data also include the State-wide average unemployment rate, population (in thousands), the percent subprime loans, and the region. Table 3 shows a fragment of the housing price index dataset and Table 4 illustrates a step-by-step protocol for Motion Chart based exploratory data analysis using the housing prices dataset. Two examples of research hypotheses that can be explored using motion charts in this setting are:

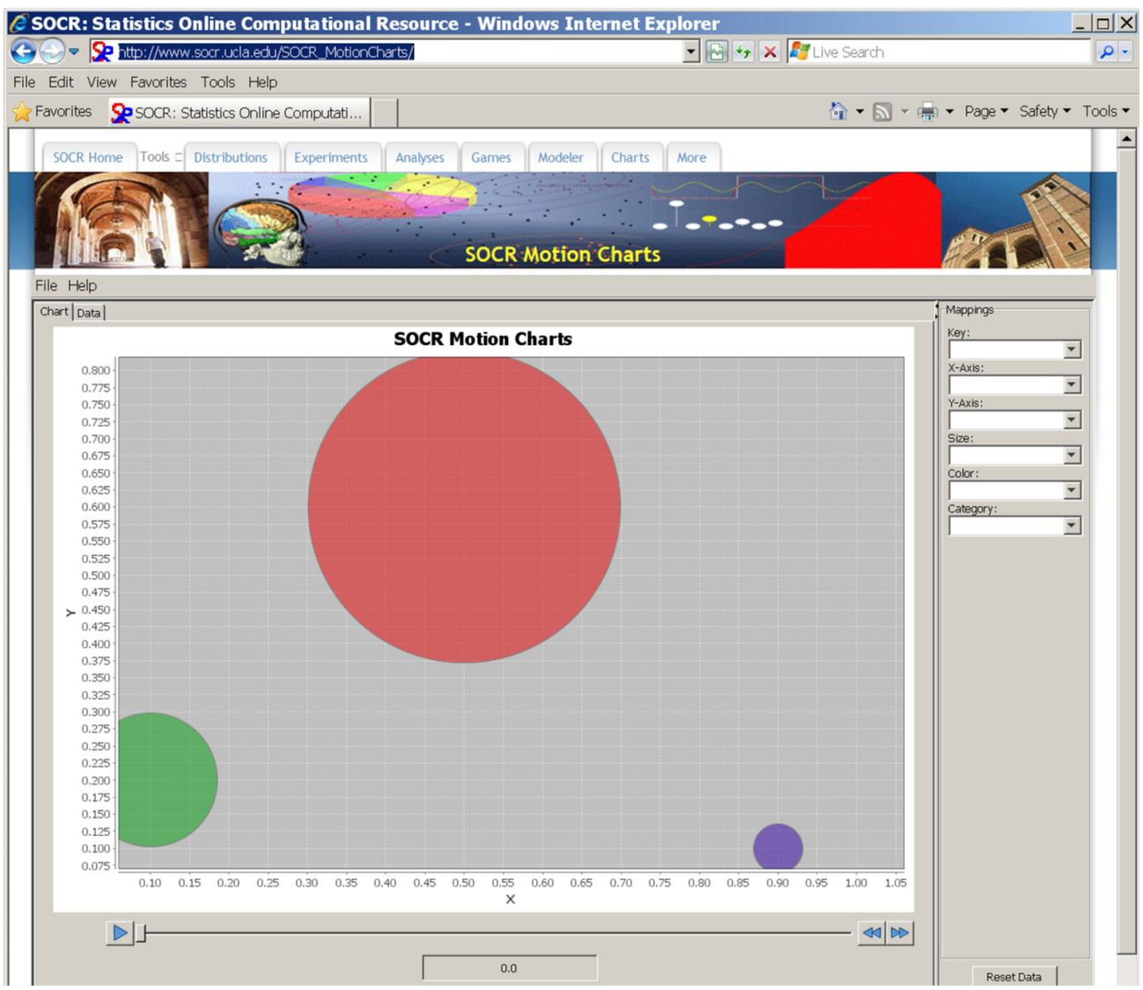
- *Q1: What is the time effect on the housing prices across states?*
- *Q2: Are there associations of the housing prices with other economics predictors like unemployment rate, population size and percent of subprime loans?*

Table 3: *A fragment of the housing price data.*

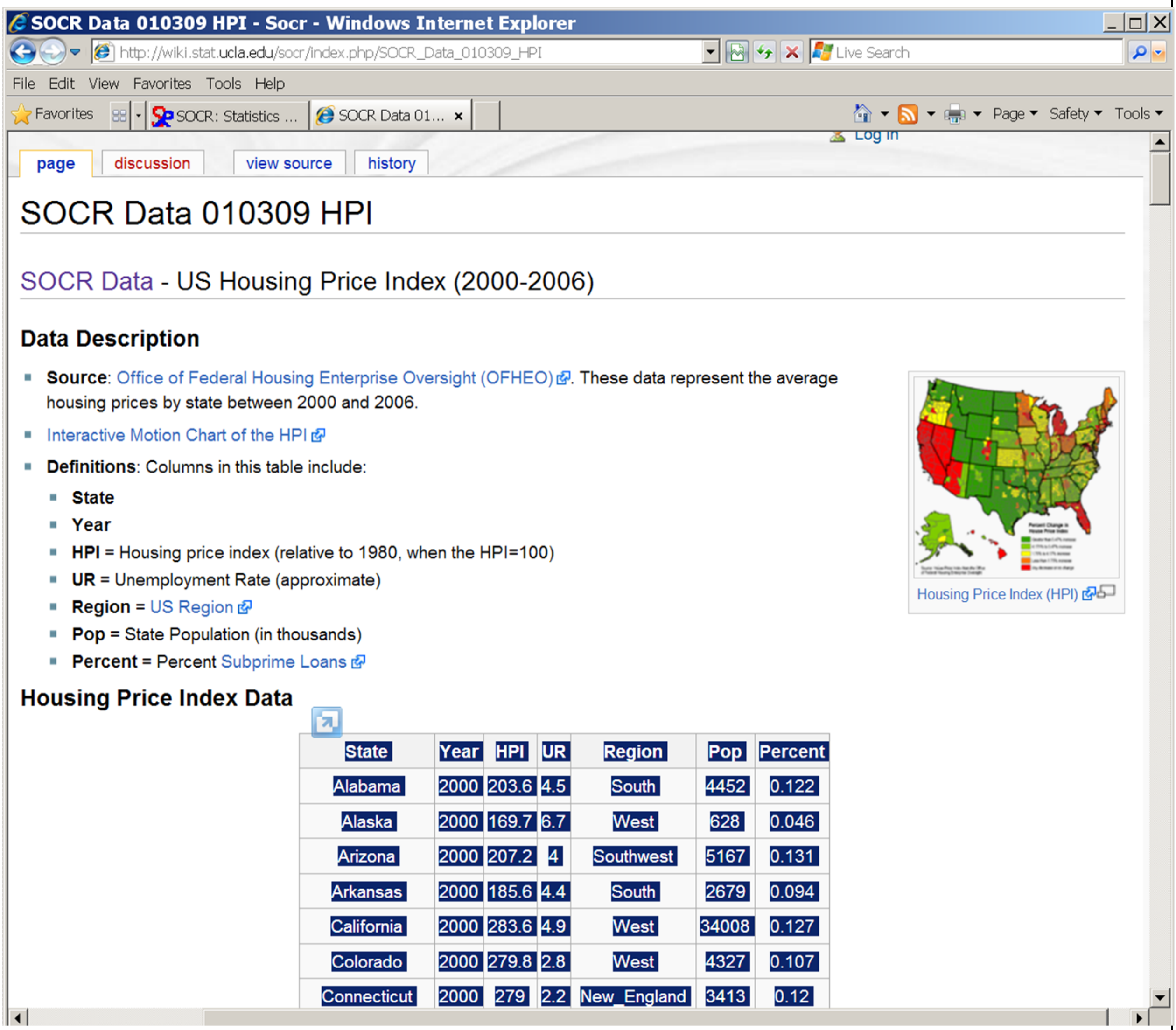
HOUSING PRICE INDEX DATA						
State	**Year**	**HPI**	**UR**	**Region**	**Pop**	**Percent**
Alabama	2000	203.6	4.5	South	4452	0.122
Alaska	2000	169.7	6.7	West	628	0.046
Arizona	2000	207.2	4	Southwest	5167	0.131
Arkansas	2000	185.6	4.4	South	2679	0.094
California	2000	283.6	4.9	West	34008	0.127
Colorado	2000	279.8	2.8	West	4327	0.107
…	…	…	…	…	…	…

The visualization of Housing Prices was particularly interesting. The housing price index data for all the states had little variance in 2000, but over the years, the variance between housing prices in different states increased significantly as shown in Table 4. In addition, at first it seemed that unemployment rate increased as the housing price index increased. However, the *Motion Chart* showed that unemployment rate began to decrease in the year 2003. The size of blobs (mapped to population) did not change significantly over the years, but the house price index did. Therefore, population and housing prices may not be correlated. A possible correlation, however, could lie between housing price index and region. This visualization implies that red states (West) and blue states (Middle Atlantic) experienced a greater increase in housing price index than other states.

Table 4: A step-by-step guide for utilization of SOCR *Motion Charts* using the Housing Price Index data.

PROTOCOL	SCREEN CAPTURE/SNAPSHOT
• Start the SOCR *Motion Charts* applet www.socr.ucla.edu/SOCR_MotionCharts (the applets starts with a simple default dataset)	

PROTOCOL	SCREEN CAPTURE/SNAPSHOT
• Identify a test dataset (URL, table, text, file, etc.)	see screen capture below

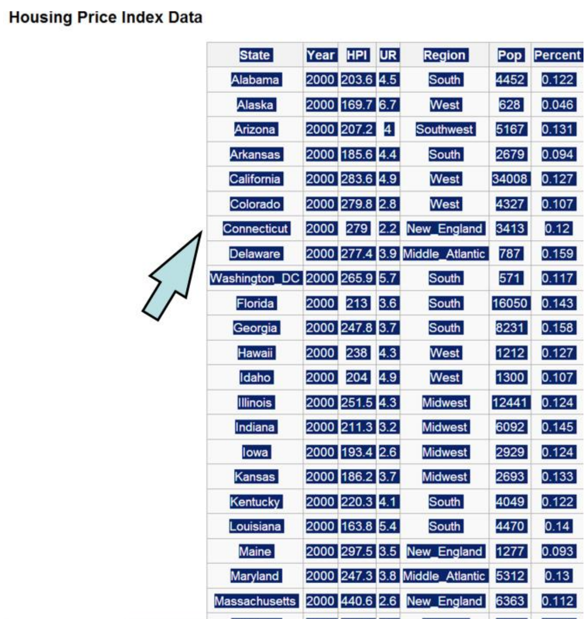


http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_010309_HPI

PROTOCOL	SCREEN CAPTURE/SNAPSHOT

- Copy the data table from your source into the Motion Charts' data spreadsheet (Data tab).

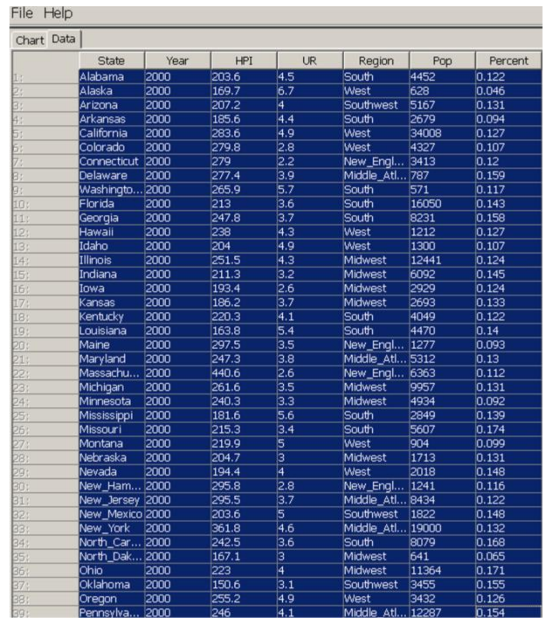
Data Source (Spreadsheet or Web page)



Housing Price Index Data

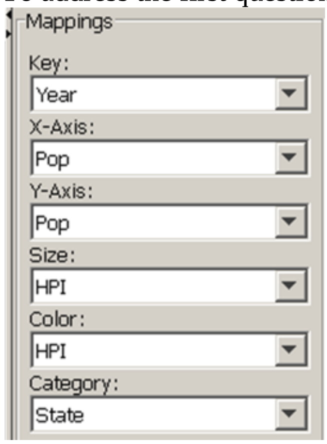
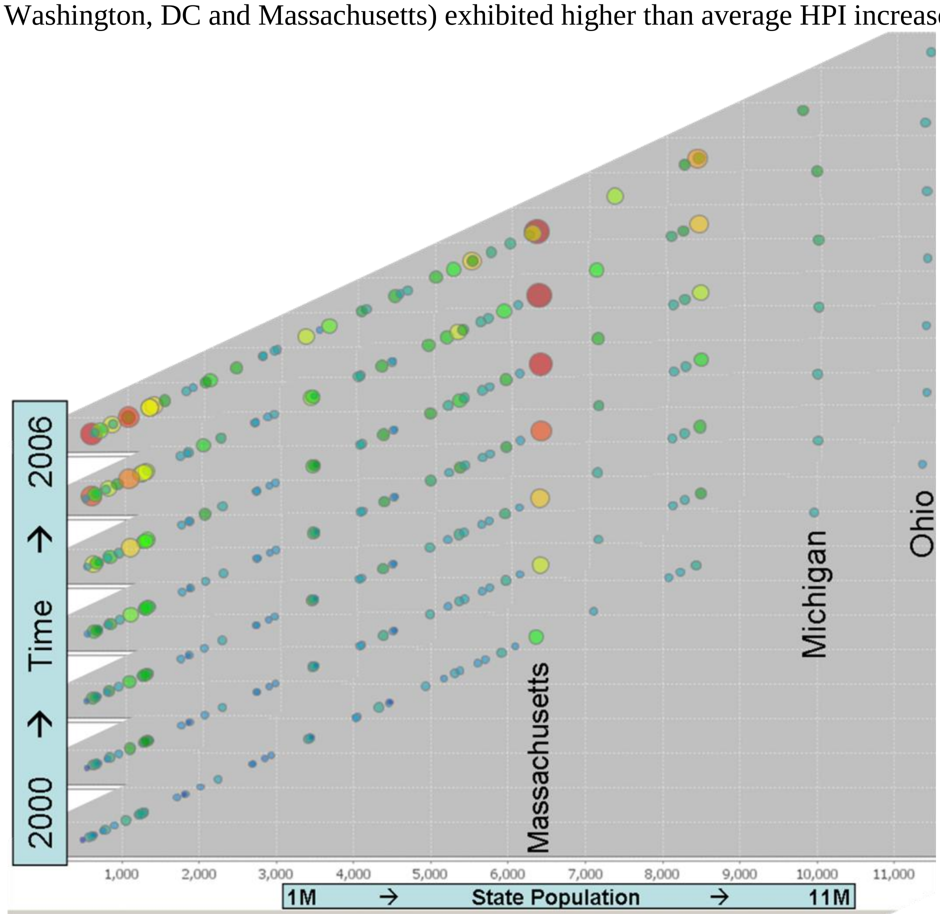
State	Year	HPI	UR	Region	Pop	Percent
Alabama	2000	203.6	4.5	South	4452	0.122
Alaska	2000	169.7	6.7	West	628	0.046
Arizona	2000	207.2	4	Southwest	5167	0.131
Arkansas	2000	185.6	4.4	South	2679	0.094
California	2000	283.6	4.9	West	34008	0.127
Colorado	2000	279.8	2.8	West	4327	0.107
Connecticut	2000	279	2.2	New_England	3413	0.12
Delaware	2000	277.4	3.9	Middle_Atlantic	787	0.159
Washington_DC	2000	265.9	5.7	South	571	0.117
Florida	2000	213	3.6	South	16050	0.143
Georgia	2000	247.8	3.7	South	8231	0.158
Hawaii	2000	238	4.3	West	1212	0.127
Idaho	2000	204	4.9	West	1300	0.107
Illinois	2000	251.5	4.3	Midwest	12441	0.124
Indiana	2000	211.3	3.2	Midwest	6092	0.145
Iowa	2000	193.4	2.6	Midwest	2929	0.124
Kansas	2000	186.2	3.7	Midwest	2693	0.133
Kentucky	2000	220.3	4.1	South	4049	0.122
Louisiana	2000	163.8	5.4	South	4470	0.14
Maine	2000	297.5	3.5	New_England	1277	0.093
Maryland	2000	247.3	3.8	Middle_Atlantic	5312	0.13
Massachusetts	2000	440.6	2.6	New_England	6363	0.112

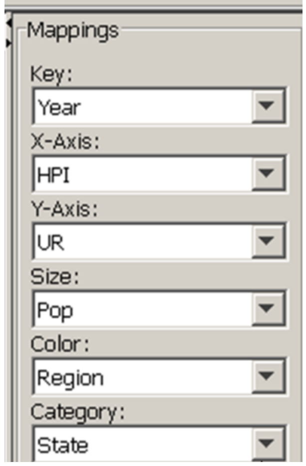
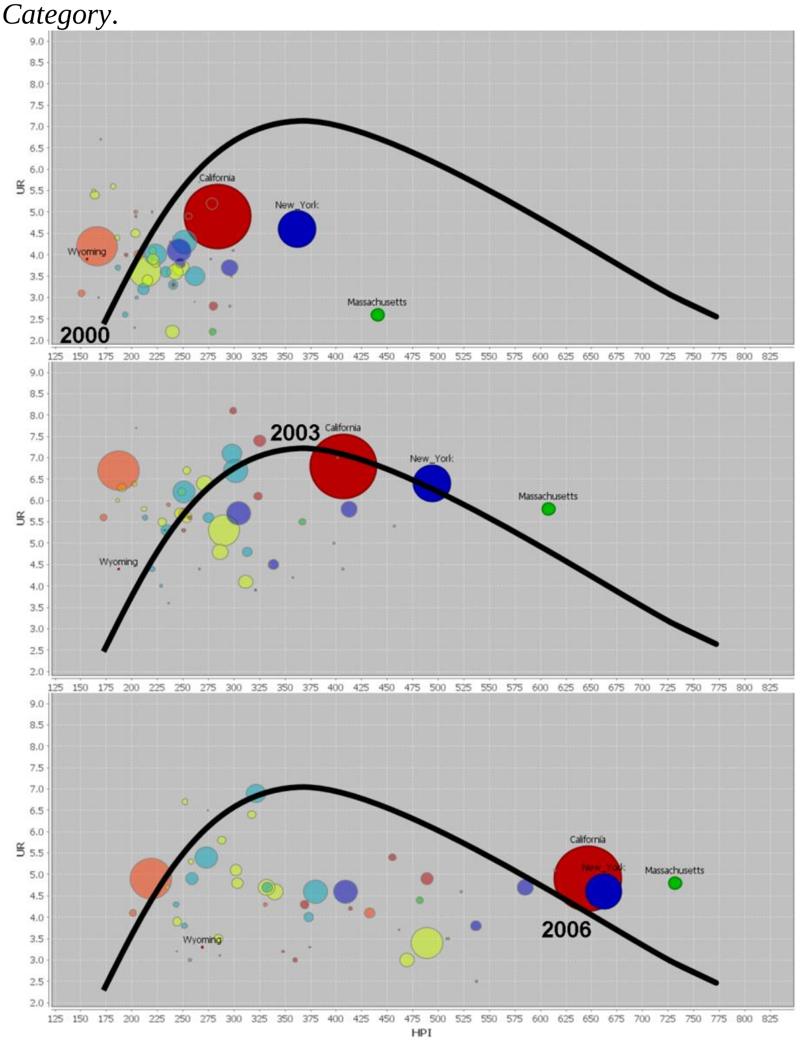
SOCR Motion Charts (Data Tab)

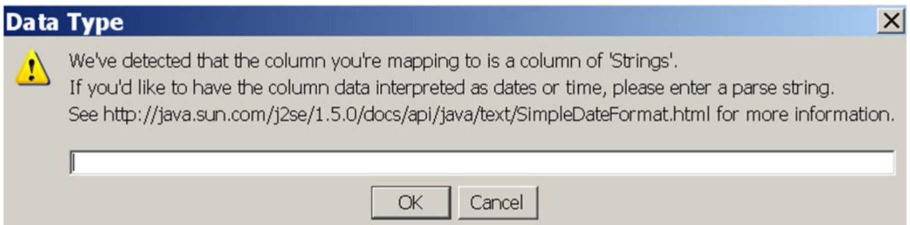
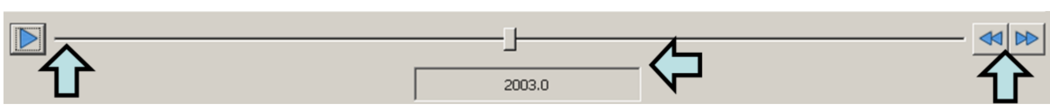


	State	Year	HPI	UR	Region	Pop	Percent
1:	Alabama	2000	203.6	4.5	South	4452	0.122
2:	Alaska	2000	169.7	6.7	West	628	0.046
3:	Arizona	2000	207.2	4	Southwest	5167	0.131
4:	Arkansas	2000	185.6	4.4	South	2679	0.094
5:	California	2000	283.6	4.9	West	34008	0.127
6:	Colorado	2000	279.8	2.8	West	4327	0.107
7:	Connecticut	2000	279	2.2	New_Engl...	3413	0.12
8:	Delaware	2000	277.4	3.9	Middle_Atl...	787	0.159
9:	Washingto...	2000	265.9	5.7	South	571	0.117
10:	Florida	2000	213	3.6	South	16050	0.143
11:	Georgia	2000	247.8	3.7	South	8231	0.158
12:	Hawaii	2000	238	4.3	West	1212	0.127
13:	Idaho	2000	204	4.9	West	1300	0.107
14:	Illinois	2000	251.5	4.3	Midwest	12441	0.124
15:	Indiana	2000	211.3	3.2	Midwest	6092	0.145
16:	Iowa	2000	193.4	2.6	Midwest	2929	0.124
17:	Kansas	2000	186.2	3.7	Midwest	2693	0.133
18:	Kentucky	2000	220.3	4.1	South	4049	0.122
19:	Louisiana	2000	163.8	5.4	South	4470	0.14
20:	Maine	2000	297.5	3.5	New_Engl...	1277	0.093
21:	Maryland	2000	247.3	3.8	Middle_Atl...	5312	0.13
22:	Massachu...	2000	440.6	2.6	New_Engl...	6363	0.112
23:	Michigan	2000	261.6	3.5	Midwest	9957	0.131
24:	Minnesota	2000	240.3	3.3	Midwest	4934	0.092
25:	Mississippi	2000	181.6	5.6	South	2849	0.139
26:	Missouri	2000	215.3	3.4	South	5607	0.174
27:	Montana	2000	219.9	5	West	904	0.099
28:	Nebraska	2000	204.7	3	Midwest	1713	0.131
29:	Nevada	2000	194.4	4	West	2018	0.148
30:	New_Ham...	2000	295.8	2.8	New_Engl...	1241	0.116
31:	New_Jersey	2000	295.5	3.7	Middle_Atl...	8434	0.122
32:	New_Mexico	2000	203.6	5	Southwest	1822	0.148
33:	New_York	2000	361.8	4.6	Middle_Atl...	19000	0.132
34:	North_Car...	2000	242.5	3.6	South	8079	0.168
35:	North_Dak...	2000	167.1	3	Midwest	641	0.065
36:	Ohio	2000	223	4	Midwest	11364	0.171
37:	Oklahoma	2000	150.6	3.1	Southwest	3455	0.155
38:	Oregon	2000	255.2	4.9	West	3432	0.126
39:	Pennsylva...	2000	246	4.1	Middle_Atl...	12287	0.154

To paste the data from the mouse buffer into SOCR Motion Charts, select the top-left most cell in the Data tab and use CTR+V or Apple+V, depending on your hardware platform.

PROTOCOL	SCREEN CAPTURE/SNAPSHOT
• Q1: What is the time affect on the housing prices across states?	To address the first question we use the following (simple) variable mapping:  This mapping allows us to focus only on the housing price index (HPI), representing both the size and color of blobs – solid discs in the graph corresponding to each state. Note that as we mapped the variable population size to the X-axis, the order of the states is based on population size (starting with Washington DC near the origin and ending with Ohio on the right). To show more detail about fewer states, we cropped the image here using the Motion Charts mouse-driven drag-and-crop functionality. To show the temporal HPI trend over time in a static image, we generated a collage of the entire motion charts animation that superimposes the graphs for all 7 time points, as shown below. The clear increase of the HPI index is visible across states in the dynamic appearance of the blobs – the increase of the blob-size and transition of the blob-color from cool (low HPI) to hot (high HPI) values. Clearly some states (e.g., Washington, DC and Massachusetts) exhibited higher than average HPI increases. 

PROTOCOL	**SCREEN CAPTURE/SNAPSHOT**
• Q2: Are there associations of the housing prices with other economics predictors like unemployment?	To investigate associations between the HPI and various predictor variables, we can generate a number of different variable mappings.  One specific variable mapping example is shown above. In this mapping *Year* is mapped to *Key*, *HPI* (Housing Price Index) is mapped to *X-Axis*, *UR* (Unemployment Rate) is mapped to *Y-Axis*, *Pop* (Population in thousands) is mapped to *Size*, *Region* is mapped to *Color*, and *State* is mapped to *Category*. 

PROTOCOL	SCREEN CAPTURE/SNAPSHOT
• Observations.	A clear parabolic trend (with respect to time) in the unemployment rate (UR) variable on the Y-axis is visible, which correlates with increase of HPI (X-axis) for all states. The parabolic model trend is shown as a right-skewed unimodal function juxtaposed on the 3 time snapshots. Clicking on any of the blobs will show their category labels (in this case State). These labels will be preserved over the motion chart animation and facilitate the easy visual tracking of specific categories of interest, e.g., states.
• Note: Mapping string variables.	 When mapping string (non-ordinal categorical) variables, the data do not have an implicit order. The following optional dialog pops up to prompt the user to provide a hash-map for "ordering" these data. Click "OK" to go with a default hash-map based on lexicographical string-ordering. If you are interested in providing a different, non-lexicographical, string ordering map for some of the variables using this dialog, the following Java documentation page shows the format your must follow to define and save a new hash-map which you can load in the applet: http://download-llnw.oracle.com/javase/1.5.0/docs/api/java/text/SimpleDateFormat.html.
• Controls.	 Use the *Play*, *Pause*, *Forward* and *Backward* buttons to control the Motion of the Chart. These controls facilitate the step-by-step exploration (forward or backward) of the motion chart animation and allow play/replay functionality for smooth exploration of the data over the entire time span.

4.2 Ice-thickness

(http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_042108_Antarctic_IceThicknessMawson)

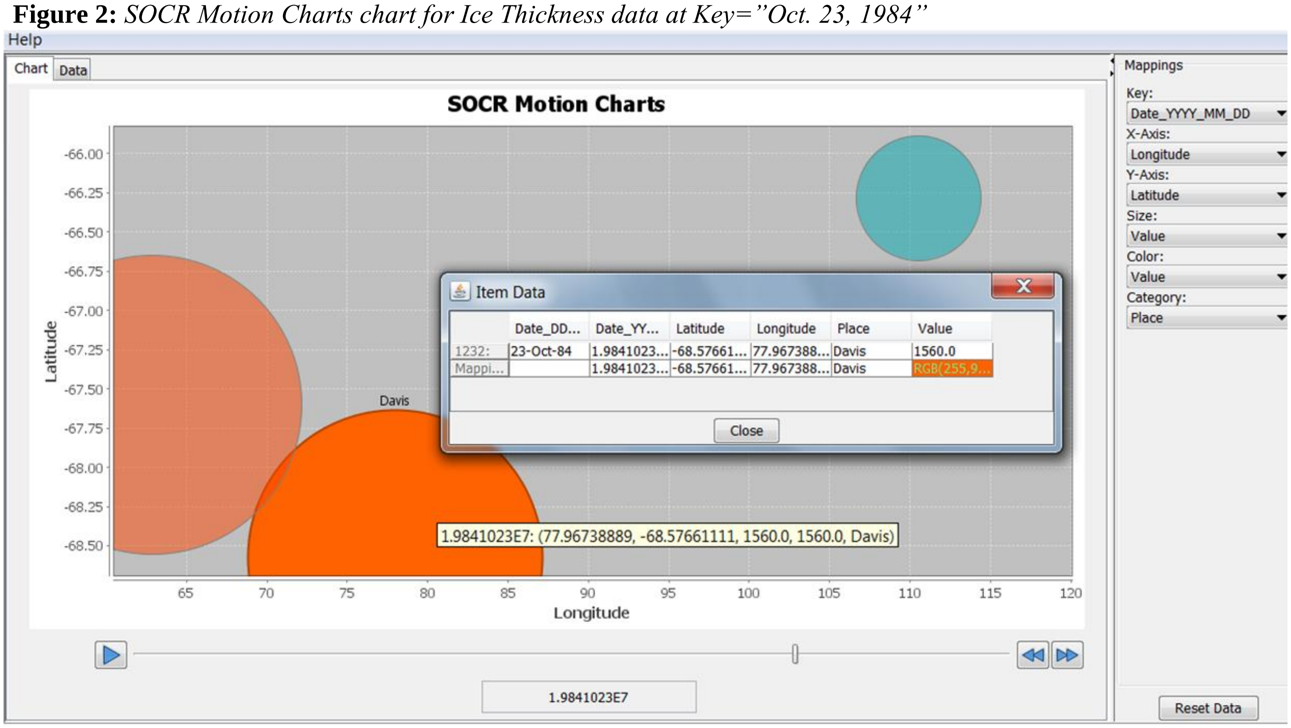
The ice thickness data were collected by Australian Antarctic Data Centers and are shown in Table 5. The measurements were taken in three different Antarctic locations: Casey, Davis, and Mawson. The ice thickness data were recorded regularly for many years. These measurements were taken on fast ice—ice that remains fastened to the land throughout the winter season. The thickness and growth rate of the ice are determined by energy exchanges with the ocean and atmosphere. When the ice thickness reaches its maximum, the optimal integration of oceanic and atmospheric conditions has been achieved. By analyzing these data, it is possible to extrapolate the effects to encompass large-scale oceanic and atmospheric processes and potentially assess global climate change. A time-based lexicographical formatting of the observation dates (using this format YYYY/MM/DD) allows the user to see 3 interesting characteristics of these data – geographic effects (between the 3 Antarctic locations), seasonal effects (Winter through Fall), and the larger year effects (1954 – 2002). These data also demonstrate the behavior of Motion Charts in the presence of missing data, or data acquired on irregular time intervals.

Table 5: *A fragment of the ice thickness data.*

ICE THICKNESS DATA					
Date DD_MM_YY	Date YYYY_MM_DD	Latitude	Longitude	Place	Value
1-Apr-54	19540401	-67.60269444	62.87380556	Mawson	1
3-Apr-54	19540403	-67.60269444	62.87380556	Mawson	90
4-Apr-54	19540404	-67.60269444	62.87380556	Mawson	200
7-Apr-54	19540407	-67.60269444	62.87380556	Mawson	280
10-Apr-54	19540410	-67.60269444	62.87380556	Mawson	340
…		…	…	…	…

The dates and interval of measurements were varied for the 3 different sites. Figure 2 illustrates one static view (October 23, 1984) of the ice-thickness data. Notice the *mapping* on the top-right side. Users may select any mapping that would satisfy their needs. In this case, we chose mapping longitude and latitude data to the 2D X and Y axes, respectively. We also set the key variable to represent time and the size and color of the blobs were mapped to the ice-thickness value. Using the *Play* mode allows automated fly through the data over the years and demonstrates an annual cycle of the ice-thickness data at all 3 locations. Mouse-over blob events trigger pop-ups with additional blob-specific information, and double-clicks on blobs extract the blob data in an external window as shown on Figure 2.

Figure 2: *SOCR Motion Charts chart for Ice Thickness data at Key="Oct. 23, 1984"*



When using the application with the Ice Thickness data, it was almost immediately obvious what months the ice was most or least thick. It was also visually clear how quickly the cycle of increasing ice thickness vs. ice melting occurred over the years. It was also clear that ice thickness peaked during late October through early November as shown on Figure 2. Although more data and further quantitative analysis are needed to make any definitive scientific conclusions, the applet helps focus the efforts of studying fast ice thickness on specific months of the year, which may suggest optimal atmospheric and oceanic conditions for collecting additional data.

4.3 Consumer Price Index, CPI

(http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_021808_CPI)

The Consumer Price Index data were provided by the U.S. Department of Labor, Bureau of Labor Statistics, and are shown in Table 6. The data represent the monthly average price for several consumer items between the years of 1980 and 2007. These data include the average prices of various consumer goods (e.g., electricity, fuel oil, gasoline, beef, chicken, fruits, etc.) Three of the interesting questions that can be asked about these data are: Is there evidence of an increase of the CPI across the years? Are there seasonal effects (e.g., Winter (food), Summer (gasoline), etc.)? Are consumer items showing different price increase rates? Mapping the Year as the Key variable and playing the motion chart should indicate if there is a clear CPI increase trend. Seasonal effects can be identified by mapping appropriate months onto the X and Y variables. Differences in the rates of CPI increase between consumer items may be shown as disproportionate size or color changes between separate blobs corresponding to different items.

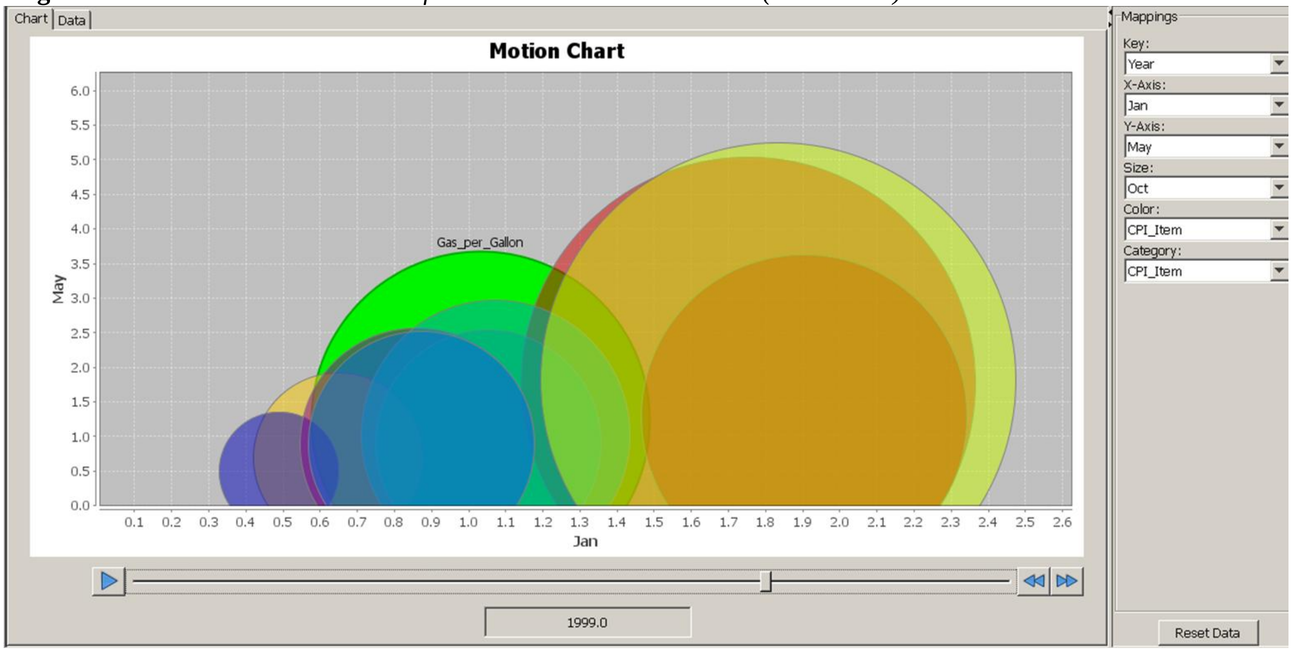
Table 6: *A fragment of the consumer price data.*

CONSUMER PRICE INDEX DATA

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	CPI Item
1980	26.698	27.701	28.15	28.433	28.96	30.931	31.513	31.569	31.755	30.89	30.381	30.967	Electricity_500_KWH
1981	31.552	32.097	32.473	32.848	33.1	34.962	35.744	36.496	36.273	34.872	34.919	34.96	Electricity_500_KWH
1982	36.006	36.086	36.741	36.799	36.438	38.047	38.403	38.471	38.265	37.388	36.655	37.074	Electricity_500_KWH
1983	37.184	37.306	37.393	37.117	37.424	39.216	39.507	39.592	39.578	38.781	38.325	38.188	Electricity_500_KWH
1984	38.6	38.832	39.066	39.487	39.539	41.635	42.406	43.35	40.801	39.441	38.935	38.626	Electricity_500_KWH
1985	38.975	39.107	39.312	39.485	39.468	41.169	41.172	41.497	40	40.25	39.752	39.754	Electricity_500_KWH
…	…	…	…	…	…	…	…	…	…	…	…	…	…

Figure 3 shows a snapshot of the SOCR *Motion Chart* generated for 1999. The mapping used to generate this chart was: *Year* to *Key*, *January* to *X-Axis*, *May* to *Y-Axis*, *October* to *Size*, and *CPI_Item* to *Color* and *Category*. Expected seasonal effect of the CPI drove the choice of the X/Y location variable mapping. Other mapping may also be appropriate for this situation, however, this specific mapping hints to the Winter vs. Summer seasonal changes in the consumer price index (e.g., gasoline consumption is known to significantly increase in the Summer months, which may be a cause for price increase). Under this mapping, seasonal price fluctuations may be reflected in the Motion Chart as blob location offsets from the line bisecting the first quadrant in the 2D plane.

Figure 3: *SOCR Motion Charts chart for Consumer Price Index data (Year=1999)*



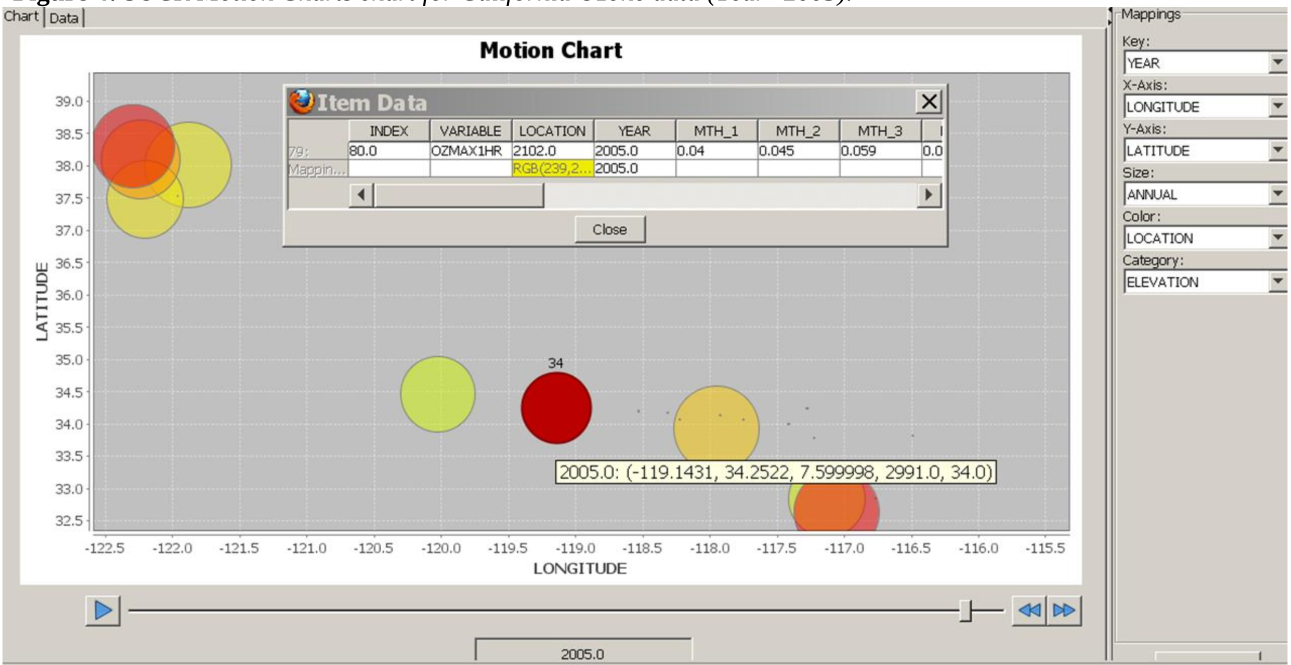
The Consumer Price Index visualization revealed an interesting trend shown in Figure 3. The price of gasoline and fuel oil remained roughly unchanged between 1983 and 1993, but then increased sharply between 1993 and 2004. The price of electricity and beef, however, increased continuously between 1983 and 2004. From this visualization, it would seem that energy costs are not correlated to fuel costs as energy costs increased continuously whereas gasoline prices increased only recently. An in-depth analysis would have to be performed to determine what the cause of the steady energy cost increase could be, and why gasoline prices changed drastically between 1993 and 2004.

4.4 The California Ozone Dataset

(http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_121608_OzoneData)

The California Ozone data include ozone levels for 20 different geographic locations throughout California accumulated over the period of 1980 through 2006 as shown in Table 1. The data are provided by The California Air Resource Board. While the complete dataset includes a number of different variables, here we demonstrate the graphs of the Maximum 1-hour Ozone Average Concentration (OZMAX1HR) variable. For every location, the latitude, longitude, and elevation were recorded. The data table includes measurements for each month of the year and an average yearly concentration for each location. Figure 4 shows a snapshot of the *Motion Chart* for these data for 2005. The chart was generated with the following mapping: *Year* to *Key*, *Longitude* to *X-Axis*, *Latitude* to *Y-Axis*, *Annual* (concentration) to *Size*, *Location* to *Color*, and *Elevation* to *Category*.

Figure 4: *SOCR Motion Charts chart for California Ozone data (Year=2005).*



The California Ozone concentration dataset visualization provided some interesting trends. The most obvious pattern revealed by the *Motion Chart* is the significant increase of ozone concentration (blob size) in more locations throughout California between 1982 and 2005 as depicted on Figure 4. In addition, the *Motion Chart* indicated that while ozone concentration increased in many locations, it did not increase significantly in areas where it was already high. This could indicate a peak ozone concentration in the atmosphere. What is perhaps most interesting is that the increase of ozone concentration began at lower elevations (blob color) and then continued to higher elevations over the years. The trend of ozone concentration increase also seems to flow from higher to lower latitudes, and from lower to higher longitudes. From the information provided by SOCR *Motion Charts*, it may be possible to trace the origin of this ozone concentration increase by following these trends and studying their origins.

5. Discussion/Conclusion

5.1 What is the challenge addressed by SOCR *Motion Charts*?

In this manuscript, we report on a new Java-based visualization paradigm for high-dimensional data – SOCR *Motion Charts*. There are two specific challenges addressed by this new interactive data rendering tool – visualization of multivariate and multispectral data, and the interactive exploration of data with temporal characteristics. The first challenge is addressed by developing a new framework facilitating the interactive display of multidimensional data by mapping ordinal, nominal and quantitative variables onto time, 2D axes, size, colors, glyphs and appearance characteristics. By harnessing the space and time dimensions, we also solve the second problem of discovery and exploration of temporal patterns, associations and trends in longitudinal data.

5.2 Benefits of using *Motion Charts*

Benefits of using SOCR *Motion Charts* for multi-dimensional data visualization become most apparent when the applet is used interactively. This applet provides a quick, convenient, and visually appealing way to find possible correlations between different variables. Previously, we would have to create several charts with different variables on the axes before we could visualize correlated data. SOCR *Motion Charts* enables the discovery of possible variable interactions and associations in a visual intuitive manner.

5.3 Potential limitations

There are several limitations of the current implementation of SOCR *Motion Charts*. As mentioned previously, the SOCR *Motion Charts* is a single-threaded applet. This imposes a limitation on the potential performance of the application on extremely complex datasets. The response of the applet may depend on the size of the dataset and may vary from computer to computer or depend on the network bandwidth and user browser settings (e.g., Java memory allocation). We have successfully tested the *Motion Charts* with datasets containing over 57,000 rows and 5 columns.

Another important limitation is the assumption that the variables in the dataset are appropriately linearly scaled and that linear transformations are appropriate for the (quantitative) data in any dimension. This, of course, is not the case with all datasets. A dataset may contain variables that are logarithmically or sinusoidally scaled. Incorrect assumptions about linearity may result in erroneous evaluation, analysis or interpretation of the data. We do, however, intend to provide the user with a means to inform the Motion Charts applet of what type of relationship should be assumed, or applied, for each variable. In essence, if the data contain variables which have been non-linearly transformed, the *flow* of the Motion Chart and the appearance of the blobs may not reflect our common linear space interpretations of spatial plots.

A minor related constraint is the potential limitation in animation capabilities. The SOCR *Motion* Charts applet is built on JFreeChart (www.jfree.org/jfreechart), an open-source Java charting package. JFreeChart is not designed for animated charts and therefore SOCR *Motion Charts* may appear to skip through some time points. A proposed solution to this problem is the injection of artificial time points through interpolation and regression. In the next version, we hope to create the option for smoother animations by adding intermediate time points or interpolated data.

While solving the problem of creating a translation to a color index, we noticed another implicit assumption that is made by the applet. The *Motion Charts* currently assume that the marginal (univariate) data are uniformly distributed. This could potentially lead to the problem where non-uniformly distributed data could have all points near the mean assigned to one color, whereas data values further away from the mean have individual colors. This is an undesirable effect as it causes the illusion that there is no change in that particular dimension for the bulk of the data. This limitation may be exacerbated by restrictions in the number of colors Motion Charts currently utilize. Currently, Motion Charts produce 512 distinct colors. While this is sufficient in most cases, if coupled with the assumption of uniform distributions, the colors assigned to the data values may not indicate the appropriate relationships between variables and data points.

5.4 Data Formats

SOCR *Motion Charts* currently accepts three types of data: numbers, dates/time, and character strings. With these data types, we feel that the application is able to handle the majority of tabular data formats. For each of these data types, we use the natural data ordering as defined by Java. While many types of data can be lexicographically interpreted as strings, it may not make sense to use lexicological ordering on all the different data types. When designing SOCR *Motion Charts*, we took this into consideration and designed the application so that it can easily be extended to provide a greater variety of interpreted data types. Thus, a developer should be able to easily extend and improve the available data types and provide additional interpretations for new types of data.

5.5 Future improvements

There are several improvements to SOCR *Motion Charts* that are planned, proposed, or under development. A significant improvement that has been planned for the next release of the application is interpolation and regression analysis. Not only would this improve the animations provided by the application, but it would (optionally) allow missing data points to be *inferred*. Currently, if a data value mapped to the x or y axis is missing, the entire blob is not displayed. If the data value that's missing is mapped to the color or size dimension, a default color and size, respectively, are used. We intend to use interpolation and regression analysis to fill in partially or completely missing data values. Alongside this feature, we also plan to add the ability to use logarithmic, exponential and polynomial models for transforming data variables.

Appendix

This appendix includes some of the technical details about the design, implementation and possible extensions of SOCR *Motion Charts*. This information may be valuable for learners, instructors and developers interested in the core *Motion Charts* engineering principles, or for extending this open-source development effort. Learners, instructors and investigators may contribute to this open-source effort by downloading, redesigning, modifying, extending and suggesting these improvements for inclusion in the core Motion Charts source code (http://socr.googlecode.com/). The SOCR Motion Charts are included in the Motion Charts package: *trunk/SOCR2.6/src/edu/ucla/stat/SOCR/motionchart*.

Why Java?
The Java programming language offers several key advantages that allowed this project to succeed. Java's most important feature is cross-platform compatibility. As with all SOCR tools, we did not wish to limit the availability of this application because of the operating system an individual or institution prefers. Java was also chosen for its framework and Graphical User Interface capabilities. The use of Java meant we could build complex GUI interfaces rapidly and with little trouble. When this advantage was combined with the Java framework, we were able to devote more focus on the actual core logic of the application and less on the representation.

How are existent SOCR Tools (including SOCR Charts/JFreeCharts) utilized?
In any successful software project, reusability is a main concern. Developing alongside other SOCR tools meant that not only did we have to create reusable code, but we had the ability to reuse everything that had already been developed. The use of existing tools allowed for a rapid development process that focused on the actual problem we were trying to solve. JFreeChart (www.jfree.org/jfreechart) an open-source Java charting package, lies at the heart of our application. JFreeChart provided the drawing and charting features required to render the visualizations of the *Motion Charts*. By harnessing JFreeChart's powerful and extensible API, we were able to create reusable, efficient, and tightly managed code.

Core design
The current Graphical User Interface is designed to maximize the available screen area for drawing the graph. The main application consists of a JScrollPane to ensure the applet cannot shrink to an undesirable state. There is a JSplitPane embedded in the main JScrollPane. The JSplitPane allows the mapping combo box controls to exist on the right while the main panel displays in the center. The mapping controls are visible at all times so that the user may immediately see the effects of changing different variable mappings—allowing them to produce the best looking graphs. The main panel consists of a JTabbedPane that hosts two tabs: Chart and Data. The Chart tab displays the actual chart, the player control, and a text field. The Data tab displays a table that accepts copying and pasting to/from Excel and any whitespace delimited data. The player control allows the user to play, fast forward, and reverse through the data. The text field shows the current key (index, time, or date) that is being displayed on the chart.

The chart is used to display several pieces of information. Blobs are used to represent a row of data. The x-coordinate, y-coordinate, color, size, and title of the blobs are all determined from the column mappings the user specifies using the mapping controls. The “Key” mapping is used to specify the temporal (or indexed) component of the data. If a key is not specified, the data is simply displayed on one graph and not split into separate graphs that can be browsed through. Currently, an x and y coordinate must be specified for a row of data to be displayed. The color is determined from a spectral color map (Dinov et al., 2006) and is related to other data rows through the natural ordering of the data in the mapped column. If a color is not specified for a particular row, or is not mapped, the blob is displayed with a gray color. Size, like color, is determined from the natural ordering of the data in the mapped column. If the size of a blob is not available, or there is no mapping for size, the border of the blob will become dashed.

The chart also allows several actions to be triggered with the mouse. Hovering over a blob causes the blob to darken and the value mapped to Category to be displayed. Double-clicking on a blob causes the data row associated with the blob to show in a table in a pop-up dialog. The table also shows the values that the data map to for chart drawing purposes in a separate row. Double-clicking on the chart itself causes all rows associated with the current key to be displayed in a pop-up table.

Implementation

SOCR *Motion Charts* was designed with simplicity, modularity, extensibility, and optimal performance in mind. The code was designed such that the GUI is separate from the core logic and data structures. This promotes modularity and allows for easier code maintenance and potentially GUI redesign. At the very heart of SOCR *Motion Charts* are three core classes and two helper classes (not including GUI classes). The three main classes—MotionTableModel, MotionDataSet, and MotionBubbleRenderer—serve to implement the core functionality of the *Motion Charts*. **Figure 5** shows how these three classes are related and how the GUI class uses them. **Figure 6** shows how these classes are related to all the classes in the *Motion Charts* package.

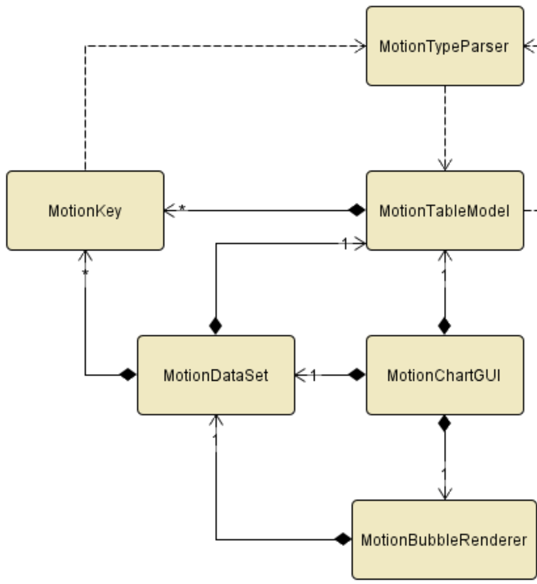


Figure 5: *The relationship of core classes with each other and with the GUI*

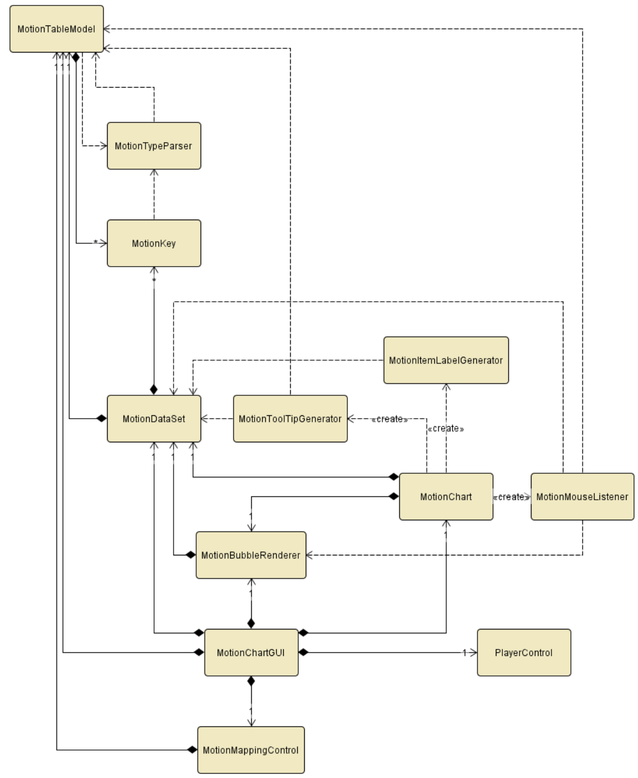


Figure 6: *The relationship between all the classes in Motion Charts.*

MotionTableModel is at the very center of SOCR *Motion Charts* and exists to host the core data structure of the applet as well as the most basic functionality. While MotionTableModel does implement the TableModel interface, for the sake of interoperability, it requires that the developer pass in a TableModel during the construction of the object. This allows the *Motion Charts* applet to work with any TableModel that a developer may deem as a more appropriate representation of their data. To implement the functionality of SOCR *Motion Charts* and optimize performance, the available data structures had to be carefully evaluated for their advantages and disadvantages. For the application, it was obvious that the data structure had to be sorted; either inherently or with a sorting algorithm. We also understood that "playing through" the data meant moving through the keys in order. This meant that we needed a map structure to associate a key with all its rows. However, the keys also had to be sorted. We also realized that, as a visualization tool, the average user is unlikely to change the data after entering them. However, should the user wish to add or delete data the data structure should support these operations. Combining all these assumptions and conclusions, we decided we could use a map and a binary tree. A Hash Map structure provides an average of *O(1)* on get and set operations, but having to sort the keys each time we wanted to access data would decrease performance significantly. A Binary Tree provides *O(log(n))* performance on get and set operations, but is inherently sorted. In the end, the tree was chosen because it was the optimal compromise. Java provides a TreeMap class that supports using keys for indexing with the performance of a binary tree. Java's implementation is based off of a Red-Black tree, which also guarantees that the performance of the tree remains constant and the tree cannot become degenerate (Cormen et al., 1990).

The TreeMap used in the MotionTableModel uses the user-mapped keys as an index and an ArrayList of rows as its mapped values. The MotionTableModel also uses several arrays to determine the columns that variables are mapped to (as specified by the user), as well as the type of data that a column contains (determined automatically). The column mappings, key map, and column classes all have appropriate accessor methods. These methods are used by the MotionDataSet to implement the majority of charting calculations. The MotionDataSet is an extension of the AbstractXYZDataset provided by JFreeChart. This class provides the appropriate dataset functionality that JFreeChart expects. The MotionDataSet is responsible for determining what different values for the x-coordinate, y-coordinate, size, and color of each blob (or row) provided by the MotionTableModel actually translates into in terms of drawing the blob on the chart.

The MotionDataSet uses a TreeSet array to store the sorted values of each column mapping and a HashMap array to store the integer mappings for Strings. The purpose of the integer mappings for strings is simply that the DataSet is expected to return a number for the x, y, and size dimensions—the color dimension also needs an integer for calculation. Therefore, we map each row of a column of strings to a number between 1 and *n*, determined by the lexicological ordering of the strings. For all dimensions, the idea of the MotionDataSet is to calculate a number that represents the relationship between any two points in the data set. This proved particularly difficult for size and color of a blob.

The color map currently used by the application requires that a zero-based integer index be used to determine the color associated with the index. The spectral color map orders the colors such that the index 0 maps to the first color in the spectrum, index 1 maps to the second color, and so on. However, the size of the color map is finite and determined at the time of creation. Since the data provided to the application can be of any range, any size, and fractional, the limitations imposed by the color map provides an interesting challenge. How do we represent the relation between any range and type of data in a memory-efficient way that can be represented with integers? The solution we proposed, while not the optimal case for all situations was using the percentage of a value in relation to the maximum value times the size of the color map (512 in our case). While this limits the number of possible distinct colors to 512, it allows any type or range of data to be mapped to colors representing the relationship of each item. The maximum value is determined simply by taking the value of the item at the tail of the TreeSet containing the values for the Color dimension. However, we noticed a problem with this solution. Negative values are mapped to negative indices which the color map does not accept. Taking the absolute value would not work as we would no longer be preserving the relationship between the values. To avoid this situation, we simply add the absolute value of the most negative value to all items as an offset to create positive integers with the same linear relationship. If there are no negative values, the offset is simply zero.

$$ColorIndex = \left(\frac{ValueOfItem + Offset}{ColumnMaxValue + Offset} \right) \times 511 \qquad \textbf{(1)}$$

The size of a blob is calculated in a similar fashion. However, the size of the blob presents two additional challenges. The size of any blob should not be zero for any blob if there is a negative value and the largest blob should not encompass the entire display area. If we used the absolute value of the most negative number as the offset, then the most negative value would translate to zero and would not have a size—this is clearly not acceptable. Also, if we used the maximum value in the column as the divisor for the percentage, the maximum value would translate to 1.0 and potentially encompass the entire display area—also unacceptable. To account for these issues we modify the calculation such that the percentage divisor is the maximum of the sum of all values in a column in a series. We define a series to be all rows associated with a particular key. Using the maximum sum as the divisor ensures that all the values for size in any series can never add to greater than 1.0. To compensate for the negative value problem, we simply add twice the offset instead of simply adding the offset; this maintains the linear relationship of values and ensures that all negative values have a non-zero positive translation.

$$SizePercentage = \frac{ItemIntensityValue + 2 \times Offset}{\max\limits_{All\ series}\left\{\sum\limits_{\text{Items in series}} IntensityValue\right\} + 2 \times n \times Offset} \qquad \textbf{(2)}$$

The final class that provides core functionality is the MotionBubbleRenderer. The MotionBubbleRenderer is an extension of the XYBubbleRenderer provided by JFreeChart. This class is responsible for the actual drawing of blobs, including what to do if data are missing or a dimension is not mapped. The MotionBubbleRenderer uses and monitors the MotionDataSet to determine how the data in a particular row should be displayed and represented as a blob. The MotionBubbleRenderer takes the x and y coordinates and translates them into screen coordinates. The color of a blob is used directly from the MotionDataSet or a default color is used if the value for a blob is not available. When calculating the actual drawing size of a blob, the MotionBubbleRenderer takes the percentage generated by the MotionDataSet and multiplies it by the length of y-axis for height (or the x-axis for width), a zoom multiplier, and a scaling multiplier. We multiply by the length of the axis to ensure that that the size of the blob is of adequate proportions to the display area. The zoom multiplier is generated from the level of zooming and is used to make sure that the size of the blobs shrinks or grows if the user zooms into or out of the graph. Without the zoom multiplier, the blobs would always remain the same size regardless of zooming. Finally, the scaling multiplier allows us to choose what percentage of the display area will be consumed by the blob.

$$BlobDiameter = SizePercentage \times AxesLength \times ZoomMultiplier \times ScalingMultiplier \qquad \textbf{(3)}$$

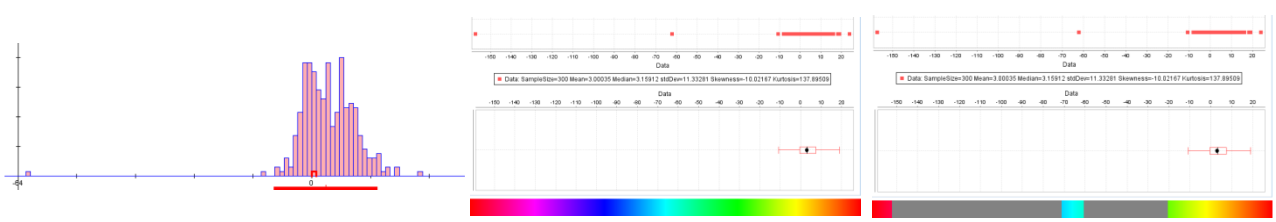
Usage and extensibility

The core Motion Chart classes all extend or implement standard Java classes, JFreeChart classes, and/or interfaces from either package. Also, almost all methods in the core classes are declared as public or protected. To a developer, this means that the Motion Chart application is easily and readily extendible and reusable. Furthermore, because the core classes are modularized, a developer can easily modify or extend just one component. For example, if a developer wishes to use a charting environment different from JFreeChart, they would could reuse MotionTableModel and possibly even MotionDataSet without major changes. Also, because the

GUI is separate from the core classes, the user interface is easily modified or redesigned. SOCR *Motion Charts* is designed with the goal of allowing a simple and intuitive method of visualizing multi-dimensional data, especially those with a temporal or ordered index component. Because of the nature of *Motion Charts*, the classes are inherently designed to provide a graphical representation of the data and not a textual representation. The application can be used to visualize the relationships in multi-dimensional data in up to four dimensions with a fifth temporal component. The design and implementation of the application allows for potentially more dimensions to be supported. The overall purpose of the application is to provide users with a way to visualize the relationships between multiple variables over a period of time in a simple and intuitive manner. Data can be copy-pasted using the mouse from any spreadsheet into the SOCR *Motion Charts* by click-selecting the top-left cell on the SOCR *Motion Charts Data* tab and then pasting the data in (CNT-V/Apple-V). Note that the first row of the data is assumed to carry the column headers, which are later used for mapping variables to graph characteristics. Blob colors may sometimes appear the same in the charts, they are, in fact, distinctly different, which is clear by double-clicking a blob to obtain a pop-up with detailed cell information. The complete source code and documentation of the SOCR *Motion Charts* is freely available online at http://socr.googlecode.com/.

Performance

Performance is a main concern with any software project. Java's Swing GUI is not thread-safe and therefore any and all modification to GUI components or their underlying models must be performed in the Swing Event Thread to avoid deadlocking. SOCR *Motion Charts* is a visual project and relies heavily on Swing components and Swing models. Unfortunately, this means that *Motion Charts* is a single threaded application. While this is not ideal for performance, especially with large datasets, it is a limitation when using Java's GUI capabilities. To improve performance with large datasets, we had to extend Java's DefaultTableModel to allow switching events on and off. By turning off events before a large operation and then turning them on when the operation is complete, we were able to significantly improve the performance of the application. The single-threaded nature of the application still imposes limitations on performance. There may be times when the user interface seems to "lock up" as calculations are being performed on excessively large datasets. This is a central concern and one which we are constantly improving.



(A) Example of a non-symmetric and dispersed dataset.

(B) Default (uniform) distribution of colors.

(C) Proposed improved color distribution based on the data quantiles.

Figure 7: A future SOCR *Motion Charts* improvement will allow input of significantly skewed data (left, A). Instead of using the default uniform color distribution over the support of the data distribution (middle, B), the new approach will employ a quantile-based color indexing to assign colors according to the true data density (right, C).

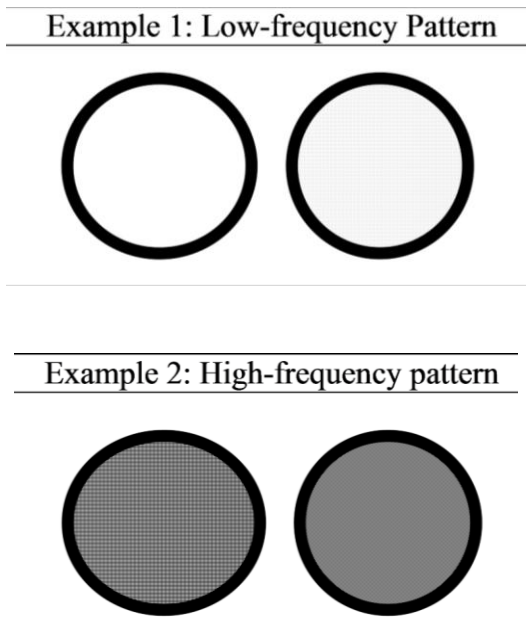
Proposed Extensions

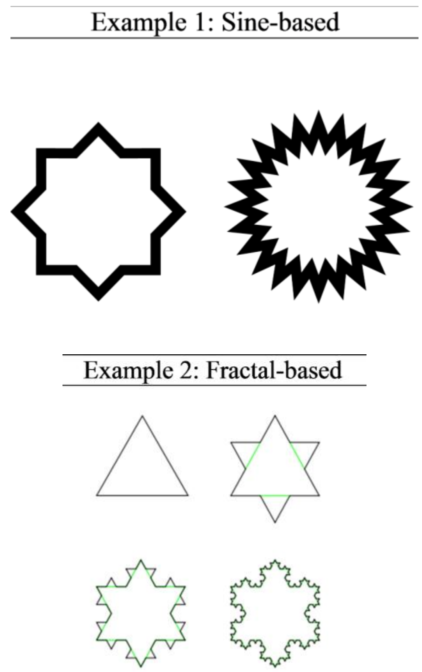
There also exist two proposed extensions to the *Motion Charts* application. The first extension/improvement provides better color handling for data that are not uniformly distributed. The proposed solution requires the use of a percentile or quantile subdivision of data intensities. Given the data set in Figure 7A, the current color indexing calculations would assign too many colors to empty spaces, and too few to the actual clusters of data as shown on Figure 7B. While we could simply create a color map of all possible color values, the problem would still exist with data that have large ranges and it would be incredibly memory inefficient. A quantile-based color index would assign more colors to areas of greater data density and fewer colors to areas of lesser data density, reducing the size of the color map and providing distinct colors to more points as illustrated on Figure 7C. Suppose we have N distinct observations and we choose to have C distinct colors, the quintile index could be calculated by T = N/C. Each quantile would contain T observations and would be assigned an appropriate color value based on their intensities.

The second proposed extension involves supporting a greater number of dimensions. Currently, the x-coordinate, y-coordinates, size, and color of the blob are the supported dimensions (excluding the temporal/indexed component). The proposed extension involves including the eccentricity, fill pattern, and stroke shape as extra dimensions. The eccentricity of the blob would indicate the directionality of the data and would be specified by a 2x2 variance-covariance matrix (Eq. 5). The fill pattern of the blob could indicate data frequency—the higher the frequency of the data, the higher the frequency of the fill pattern as shown on Figure 8A. Finally, the stroke shape of the blob refers to the oscillatory complexity of the blob boundary. This dimension could be mapped to any ordered value. The higher the value in relation to other values in the column, the greater the boundary complexity would be, see Figure 8B. The boundary complexities could be determined through sine-wave based methods (Eq. 6) or fractal based methods as shown on Figure 8B.

$$\sum = \begin{pmatrix} \sigma_X^2 & \sigma_{X,Y}^2 \\ \sigma_{X,Y}^2 & \sigma_Y^2 \end{pmatrix} \qquad \textbf{(5)}$$

$$\begin{cases} X = X(s) * \left[1 + \cos\left(\frac{2\pi \times s}{L} \times Shape \right) \right] \\ Y = Y(s) * \left[1 + \cos\left(\frac{2\pi \times s}{L} \times Shape \right) \right] \\ 0 \le s \le L \end{cases} \text{, where L = curve length.} \qquad \textbf{(6)}$$





(A) Using blob appearance to encode variable information.

(B) Employing boundary shape characteristics to represent multivariate data.

Figure 8: Another future SOCR Motion Charts improvement will provide blob-appearance cues. For example, the blob fill pattern may indicate data frequency (left, A), and the shape of the blob boundary may represent another variable (right, B).

Acknowledgments

We are thankful to all SOCR faculty, students and staff for their help, motivation and support for the development, deployment and validation of SOCR *Motion Charts*. All SOCR efforts are funded in part by NSF grants 0716055 and 0442992, and NIH grant U54 RR021813. We are indebted to the constructive critiques and valuable recommendations of the JSE editors and the anonymous reviewers.

References

Cohen L., Manion L., Morrison K. (2007). Research Methods in Education. 6th ed. Routledge.

Cormen T., Leiserson C., Rivest R. (1990). Introduction to Algorithms MIT Press.

Dinov I. (2006). Statistics Online Computational Resource. *Journal of Statistical Software* 16:1-16.

Dinov I., Christou N. (2009). Statistics Online Computational Resource for Education. *Teaching Statistics* 31:49-51.

Dinov I., Sanchez J., Christou N. (2008). Pedagogical Utilization and Assessment of the Statistics Online Computational Resource in Introductory Probability and Statistics Courses. *Journal of Computers & Education* 50:284–300.
DOI: http://doi:10.1016/j.compedu.2006.06.003.

Dinov I., Christou N., Gould R. (2009). Law of Large Numbers: the Theory, Applications and Technology-based Education. *Journal of Statistical Education* 17:1-15.

Dinov I., Valentino D., Shin B., Konstantinidis F., Hu G., MacKenzie-Graham A., Lee E., Shattuck D., Ma J., Schwartz C., Toga A. (2006). LONI Visualization Environment. *Journal of Digital Imaging* 19:148-158.

Grossenbacher A. (2008). The globalisation of statistical content Statistical Journal of the IAOS: *Journal of the International Association for Official Stati*stics 25:133-144.

Le Gal G., Righini M., Parent F., Van Strijen M., Couturaud F. (2006). Diagnosis and management of subsegmental pulmonary embolism. *Journal of Thrombosis and Haemostasis* 4:724-731.

Safran C., Bloomrosen M., Hammond W., Labkoff S., Markel-Fox S., Tang P.C., Detmer D.E., panel W.i.f.t.e. (2007). Toward a National Framework for the Secondary Use of Health Data: An American Medical Informatics Association White Paper. *Journal of the American Medical Informatics Association* 14::1-9. DOI: 10.1197/jamia.M2273.

Vermylen J. (2008). Visualizing Energy Data Using Web-Based Applications. Trans. American Geophysical Union 89:Fall Meet. Suppl., Abstract ED53A-0588.

Zimmer J., Monroe M., Qian W.-J., Smith R. (2006). Advances in proteomics data analysis and display using an accurate mass and time tag approach. Mass Spectrometry Reviews 25:450-482.

Jameel Al-Aziz
Statistics Online Computational Resource
Department of Computer Science and Engineering
University of California, Los Angeles
Los Angeles, CA 90095
Email: jalaziz@ucla.edu

Nicolas Christou
Statistics Online Computational Resource
Department of Statistics
University of California, Los Angeles

Los Angeles, CA 90095
Email: nchristo@stat.ucla.edu

Ivo D. Dinov
Statistics Online Computational Resource
Department of Statistics
Laboratory of Neuro Imaging
8125 Mathematical Science Bldg
Los Angeles, CA 90095, USA
Tel. 310-825-8430
Fax: 310-206-5856
http://www.stat.ucla.edu/~dinov
dinov@stat.ucla.edu
