Eurographics Conference on Visualization (EuroVis) 2013
B. Preim, P. Rheingans, and H. Theisel
(Guest Editors)

Volume 32 (2013), Number 3

ExPlates: Spatializing Interactive Analysis to Scaffold Visual Exploration

W. Javed and N. Elmqvist

Purdue University, USA

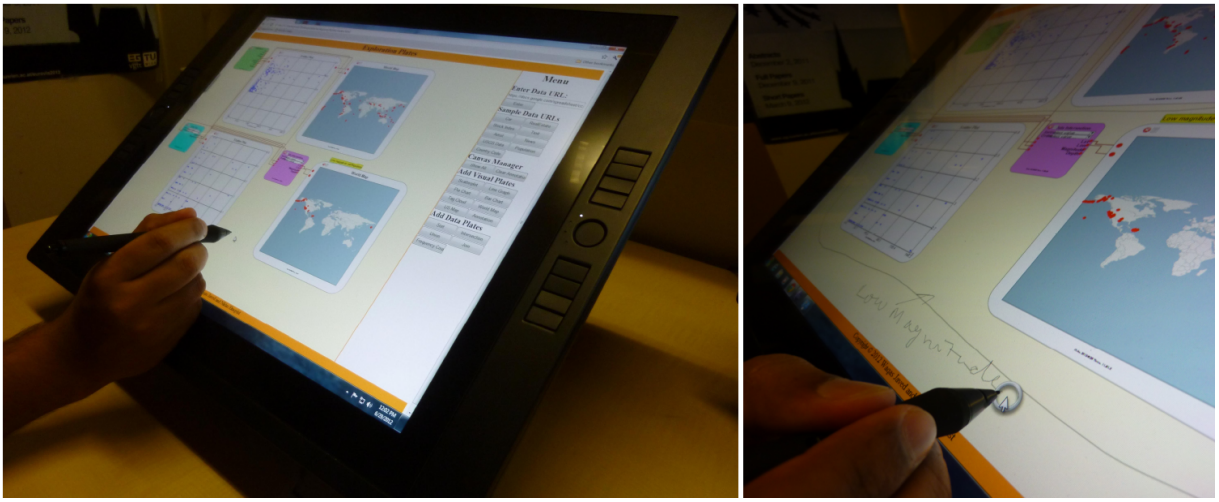

Figure 1: *An instance of visual exploration using the ExPlatesJS system. After performing an initial exploration, the user is annotating different exploration states using the freehand annotation feature supported by the system.*

Abstract

Visual exploration involves using visual representations to investigate data where the goals of the process are unclear and poorly defined. However, this often places unduly high cognitive load on the user, particularly in terms of keeping track of multiple investigative branches, remembering earlier results, and correlating between different views. We propose a new methodology for automatically spatializing the individual steps in visual exploration onto a large visual canvas, allowing users to easily recall, reflect, and assess their progress. We also present a web-based implementation of our methodology called ExPlatesJS *where users can manipulate multidimensional data in their browsers, automatically building visual queries as they explore the data.*

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

1. Introduction

Exploratory data analysis (EDA) [Tuk77] is the process of collecting insights about a dataset without using a statistical model or an a priori hypothesis. In other words, EDA is a data-driven approach to analyzing large and complex data, and often uses visualization to facilitate the process [Cle94]. For example, in PRIM-9 [TFF88], John Tukey demonstrates the power of creating interactive visual representations of an unknown dataset when starting to analyze it. This activity is often known as *visual exploration* [Kei02], and is the ap-

proach most users take when utilizing a visualization tool to analyze data—using vision to think [CMS99].

However, exploratory data analysis, even when aided by visual means, often places a high cognitive load on the analyst in terms of memory, perception, and reasoning [TC05]. In this paper, we propose a new methodology for automatically *spatializing* the individual steps in visual exploration onto a large visual canvas, allowing users to easily recall, reflect, and assess their progress. In essence, instead of using a single viewport with a visual representation that changes as the user selects, filters, and navigates in the data, we cause every interaction with a visualization to spawn a new visualization where the desired change—such as a filter, annotation, or change of visualization technique—has been enacted. Similar to earlier work on chunking interaction [HMSA08, KF88], the methodology distinguishes between interactions that cause the visual representation or the underlying data to change—*mutating operations*—and those that only change its viewport or formatting—*invariant operations*. The former (mutating) operations are those that warrant spawning a new immutable visualization instead of changing an existing one. The benefit of this radical approach to spatializing visual exploration is to *externalize* [SR96] not just the data being visualized, but the exploration process itself used to analyze the data. Furthermore, the visualization viewport itself *becomes* the history; users can disregard its presence when it is not needed, or choose to overview or navigate through it when it is.

To validate our new methodology, we present a web-based visualization system called ExPlatesJS where users can explore multidimensional data in their browsers, automatically building visual queries as they explore the data (Figure 1). The ExPlates system is based on creating and connecting *exploration plates* on an infinite, zoomable and pannable *exploration canvas*, where the plates can either signify datasets, analytic operations, or visual representations. The user explores the data by adding datasets and visualization plates to the canvas, and then wiring them together. Datasets, analytic operations, and visualization plates can be annotated using text or free-hand drawing, as can the visual canvas itself (see the pen-based interaction used in Figure 1). Mutating user operations performed on a visualization plate automatically spawns a new visualization plate, often accompanied by an analytics plate that performs the actual operation desired by the user. Furthermore, analytic plates can be manually added to perform higher-level computation on the data, such as calculating descriptive statistics, characterizing data distribution, or geolocating addresses.

2. Background: Spatializing Visual Exploration

Using visualization to develop insight about data is an iterative process, where an analyst progressively interacts with a data visualization to develop, confirm, or reject hypotheses about the underlying data [Kei02].

2.1. Cognitive Load of Visual Exploration

Visual exploration is widely acknowledged as a complex, cognitively taxing, and potentially time-consuming process [JKMG02, GZA06]. Shneiderman's information seeking mantra (*"overview first, zoom + filter, details on demand,"*) [Shn96] does not cover the iterative behavior of the sensemaking process [RSPC93]. In reality, the process consists of a long sequence of interactions and decisions where the results of one step informs the next step [JKMG02, GZA06]. For this reason, the analyst often experiences a high cognitive load when performing visual exploration [APM*11]. We highlight the following factors that affect this cognitive load:

- **Perception:** The analyst's perceptual skills are taxed by visually complex images and the need to split attention between several parts of a representation. Visual clutter can arise from both a small set of complex objects, as well as from a large set of simple objects.
- **Memory:** Many sensemaking tasks require the users to remember past state and previous results in order to make sense of new state and avoid repeating exploration.
- **Reasoning:** Comparing, correlating, and combining results from multiple visualization states is a difficult undertaking even for a trained analyst [WSP*06].

2.2. Modeling Visual Exploration

Visual exploration is characterized by the visualization system changing state over time as the user interacts with it, such as filtering, reconfiguration, and navigation [YaKSJ07]. To model visual exploration, we therefore need to model the states that a visualization goes through during this process. Based on Card et al.'s [CMS99] pipeline, we define a *visualization state* as a tuple of three attributes (D, M, V), where:

- D represents data transformations;
- M represents the visual mapping; and
- V represent view transformations (navigation).

Whenever an interaction operation changes one or more of these three attributes, we say the *visualization state has changed*. In order to perform different analytical tasks, the analyst needs to keep track of different visualization states, generated in response to interactions conducted over an analysis session. For example, it is common to compare a new visualization state with a previous state in order to confirm or reject a hypothesis about the underlying data. Similarly, through the course of an analysis session, many times the analyst needs to go back to some previous state and start a new exploration branch and compare that with previous exploration. However, the majority of current visualization tools do not provide support for interacting with multiple visualization states that were generated at different times.

2.3. Scaffolding Visual Exploration

To remedy this state of affairs, we propose a new interaction methodology for *spatializing* exploration where new visualization states, dynamically created by interaction operations, do not replace old ones, but are instead added to the same visual substrate. In other words, instead of modifying a visualization state in-place, the original state is left unchanged and a new visualization state with the required modifications is created. This spatialization should be performed so that:

1. Different visualization states can be compared;
2. Relations between visualization states can be seen; and
3. Interaction apply to any of the visualization states.

However, depending on the underlying visual representation, not all changes to a visualization state are necessary to capture. For example, formatting changes to the marks in a scatterplot may not be important to the overall exploration. For this reason, we distinguish between interactions that cause the visual representation or the data to change—*mutating operations*—and those that only enact cosmetic changes—*invariant operations*. The former (mutating) operations are those that warrant spawning a new immutable visualization instead of changing an existing one. This is in spirit with work on chunking interaction [HMSA08, KF88]. The choice of mutating and invariant operations varies based on the visual representation and the analysis tasks.

We believe that this modality of interacting with multiple exploration states can help address the cognitive challenges discussed above by *externalizing* [LS87, Nor93, SR96]—i.e., off-loading, re-representing, and constraining—not just the data being visualized, but the exploration process itself.

3. Design Space

Visual exploration has been a very active research topic in the visualization community, and much results exist on its fluid, iterative, and progressive nature [GZA06, HS12, JKMG02]. The design space of methods for scaffolding visual exploration includes the following topics;

- **Graphical histories** for capturing, storing, and visually representing interaction over time;
- **Exploration trees** that go beyond mere histories to capture and visualize branching exploration;
- **Storytelling and communication** approaches that utilize visualization for narrative purposes;
- **Data provenance** mechanisms for managing data sources and how visualizations were created; and
- **Composition and data flow** for visualization, where complex queries can be built from simple components.

3.1. Graphical Histories

History mechanisms are common in human-computer interaction (HCI), primarily due to their relation to undo and redo operations. Visualization tools generally do not allow for editing data and thus typically have no undo and redo, so the field has traditionally not seen the same need for capturing and managing the state of the visualization process as the HCI community. However, histories have also been used for capturing navigation, such as in a web browser, suggesting that there is ample opportunity to utilize history mechanisms even for situations where no content editing exists.

Heer et al. [HMSA08] carefully review the design space of interaction histories; we defer to their treatment except to highlight some of the examples that are relevant to visualizing history. Today's visual applications are particularly amenable to *graphical histories* that maintain not just previous interaction states, but also show them using a graphical summary [KF88]. This is most commonly done using a thumbnail image of previous state [Ma99, HMSA08]. In recent work, Heer et al. [HMSA08] propose a comic strip history for Tableau that augments the existing interface of the tool using thumbnail images of previous visualization state.

- **D1:** Interaction histories are important, and a graphical approach is especially amenable to visualization tools.

3.2. Exploration Trees

A linear sequence of previous states is not desirable for visualization purposes, where an abandoned branch may either signify a hypothesis that did not pan out or an insight that the user found at the end of that branch. For this reason, visualization tools tend to make the leap to *exploration trees* that capture the branching nature of visual exploration. One such tool was GRASPARC [BPW*93], a problem solving system explicitly built to leave an "audit trail" that can then be interactively visualized using a history tree. Derthick and Roth [DR01] show how such a "branching time model" can be utilized to off-load user memory and allow for comparing results across time and scenarios, whereas other approaches take a sequential view of the hierarchy [SvW08].

- **D2:** An exploration history should be represented as a tree, allowing users to navigate and recall all past states.

3.3. Storytelling and Communication

Visualization is also a great tool for telling a story and communicating insights to an audience, but this has not traditionally been a focus of the visualization community. Thomas and Cook in 2005 highlighted the production, presentation, and dissemination of insights gained during analysis as one of the grand challenges of visual analytics [TC05]. Much work on communication and storytelling has been proposed since. Most recently, Segel and Heer [SH10]—riding on these trends as well as the new wave of infographics that are using visual mechanisms to communicate stories to a broad audience—surveyed the design space of narrative visualization and proposed distinct genres, design guidelines, and storytelling mechanisms that are useful for this purpose.

- **D3:** Visualizations should ideally not only be designed for analysis, but should also enable communicating insights.

3.4. Provenance for Visualization

Provenance aspects of data visualization states that it is not sufficient to gain insights, or even to communicate them to an external audience, but that this audience must also be able to satisfy themselves as to the accuracy, sources, and process followed in attaining the insights. Groth and Streefkerk [GS06] propose a model for recording the provenance of visual exploration and allowing users to annotate their analytical process. Notable among provenance-maintaining systems is VisTrails [BCS*05], where the specification of the specific visualization pipeline—the *visualization trail*—used to arrive at a particular result not only concisely captures a configuration that can be executed and reused, but it also provides a record of how the results were generated.

- **D4:** Provenance allows users to determine data sources and how a visualization was created.

3.5. Composition and Data Flow

While monolithic visualizations suffice for representing simple datasets and small problems, more complex and heterogeneous data often require combinations of several different visualization techniques [JE12]. Highly dynamic visualization dashboards exist, such as Dashiki [McK09]. However, while powerful, building a dashboard requires some in-depth knowledge, and the resulting dashboards convey little about their structure and provenance, i.e., *how* they were built.

A competing approach that puts the structure and provenance of data as its first priority has instead been *data flow systems* where the user constructs a query using a visual grammar of components that are wired together to form a data pipeline. Examples include the Sandbox [WSP*06], which is a semi-structured analysis space for intelligence analysts; the DataMeadow [EST07], where multidimensional visual queries are built by stringing together starplots on an infinite canvas; and, most recently, GraphTrail [DRL*12], where multimodal and multivariate graphs are explored through a sequence of visual components stringed together on a reasoning space. However, so far, very few of these systems have been designed for web-based use. Impure (`http://www.impure.com/`), a visualization editor, is one of a few exceptions of web-based data flow system.

- **D5:** Visual data flow systems allow end-users to build visualizations tailored to specific problems, data, and tasks.

4. ExPlates: Technique

Based on spatializing exploration states, we developed a flow-based visual exploration technique that we call *Exploration Plates* (ExPlates). ExPlates is designed to seamlessly spatialize visual exploration states as a result of user interaction (D1, D2), and each state is then recognized as an independent visual structure that we call an *exploration plate*. Over the course of an exploration session, the user creates multiple exploration plates, where each plate specifies a data operation or a visual representation. Correspondingly, in order to differentiate between these transformations, the ExPlates technique introduces two types of exploration plates:

- *Data plates* are used to highlight data transformations performed on the underlying data as a result of different data operations, such as sorting, joining, or intersecting data.
- *Visualization plates* represent a visual transformation from data to a visual representation inside its extents.

Exploration plates are automatically laid out on an infinite *visual canvas*. In order to highlight the flow of data between different plates, the ExPlates technique relies on the notion of *data wires* and *data anchors* (D5). Data wires represent the connection between any two plates, with the metaphor that data flows along the wire similar to an electrical current flows along a copper wire. Data anchors are the ports on the source and destination plates where the wires are connect, similar to pins on an electronic component. Plates can also be deleted when they are no longer needed. Further, plates can be annotated to communicate and recall findings particular to each plate (D3, D4). Moreover, to maintain provenance (D4), the technique saves the state of the visual canvas and so that it can be retrieved later for further exploration. Since the ExPlates technique is intended for casual users [PSM07], the technique is also designed for use in a web browser.

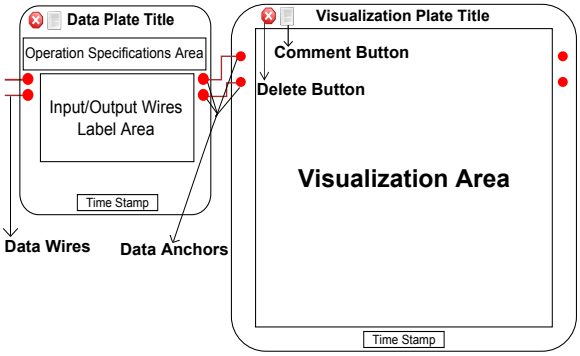

Figure 2: *Illustration of different ExPlates components.*

5. ExPlates: System

Our implementation of Exploration Plates is called ExPlatesJS, and is a web-based visualization system for spatializing visual exploration. Below we give more details.

5.1. Implementation

We have implemented ExPlatesJS, a web-based system, using JavaScript and SVG. We use the Google Data Source

API to access online data repositories such as Google Documents, RSS feeds, Atom feeds, XML, and CSV files. The architecture is designed to make it easy to extend with new visualization and data plates, even using other SVG-based visualization libraries such as D3 [BOH11], the JavaScript InfoVis Toolkit, Raphaël, and Processing.js. The current implementation uses the Raphaël toolkit for rendering.

5.2. Visual Canvas

ExPlatesJS consists of a visual canvas on which different plates are laid out. The canvas is infinitely zoomable and pannable using a left drag (panning) and mouse wheel (zooming). This allows the user to zoom in on different exploration plates during an analysis session, avoiding visual clutter. Besides user-controlled navigation, the canvas also supports predefined animated navigations, e.g., to show all the exploration plates through an animated zoom-out.

5.3. Layout Algorithm

We use a grid-based layout algorithm for the visual canvas, where the width of a cell inside this grid is dynamically adjusted to the width of each plate, while the height of each cell is fixed to the height of the largest plate in that same grid row. The rationale behind these design decisions was the idea of developing multiple exploration branches where a horizontal row signifies a single exploration branch. This is similar to previous history trees approaches [BPW*93,DR01,SvW08]. Furthermore, to make space for data wires, our layout algorithm leaves padding between adjacent cells in the grid.

New plates are generated either by the user selecting a particular type of data or visualization plate from a menu, or as a result of a user performing a visual query inside a visualization plate. The latter plates are always placed in the next position to the right on the same row as the plate inside which the query was performed. On the other hand, whenever a new plate is generated by the user selecting a particular plate from the menu, its position inside the canvas is calculated either by the user or automatically by the layout algorithm. In case of a user-defined placement of the new plate, the user selects an existing plate using a mouse click to indicate that the new plate is to be placed in the next right cell of that row. The selected plate glows to make this selection visible. If the next right cell of the selected plate is already occupied, the system shifts all those plates to the right using a smooth animation to make space for the new plate. If the user does not specify the position for the new plate, it is placed at the bottom left corner of the grid, effectively starting a new exploration branch.

Furthermore, users can drag and drop plates to change their position within the grid. Once the user makes a change, the final position of the plate inside the grid is calculated based on the nearest cell to its top left corner and, if required, any other plate is shifted left or right to keep the layout compact. Plates can be deleted by clicking on a delete button, placed near its top left corner. Whenever an existing plate is deleted, all the plates on its right are shifted to left to fill its space. If the deleted plate was the only plate in that horizontal grid, all the plates below that are shift one row up. To avoid any inconsistency in the data flow, the user can only delete plates with no output data wires; furthermore, whenever a plate is deleted, its input data wires are also deleted.

5.4. Data Anchors and Data Wires

Data wires are drawn using a thin dark line, signifying the flow of values for a single data dimension from one plate to another. Each data wire starts and ends at the data anchors of the connected plates. When the user hovers the mouse on a data wire, it is highlighted and a tooltip shows the label of the data dimension associated with that wire.

Data anchors are represented using red circles at the left or right borders of an exploration plate. Data anchors on the left border of a plate are input data anchors, each connected to a unique input data wire, while data anchors on the right border are output data anchors that may or may not be connected to an output data wire. An input data anchor acts as the destination of only one input data wire; however, an output data anchor may act as the source of multiple data wires. Each data anchor stands for a particular data dimension and the label of that dimension is shown near that anchor.

The user creates a new data wire by starting a drag operation inside any output anchor, making that anchor the source of the new wire. A rubberband representation of the wire then follows the mouse pointer, and if the user releases the drag operation inside an existing plate, a new input data anchor is created on that plate. If the user ends the drag operation on empty space, the new wire is simply deleted from the canvas. The user can disconnect a wire by performing a drag operation starting on one of its connected anchors, and either release it on another anchor to reconnect it, or on empty space to delete it. Whenever the user brings the mouse pointer inside any data anchor, its associated wire is highlighted, thereby identifying the source and destination of a particular wire. Furthermore, by clicking on a input anchor, the user can animate the viewport to the source of its associated input data wire. This feature is particularly important when the source and destination plates are located far apart.

5.5. Data Plates

ExPlatesJS currently supports seven data plates. Each type is color-coded to help the user distinguish them. Figure 2 shows the general structure of a data plate, where the title describes the operation performed by that plate, and the operation specification area is used to display different options related to its data. A time stamp at the bottom of each data plate indicates the date and time when that plate was created.

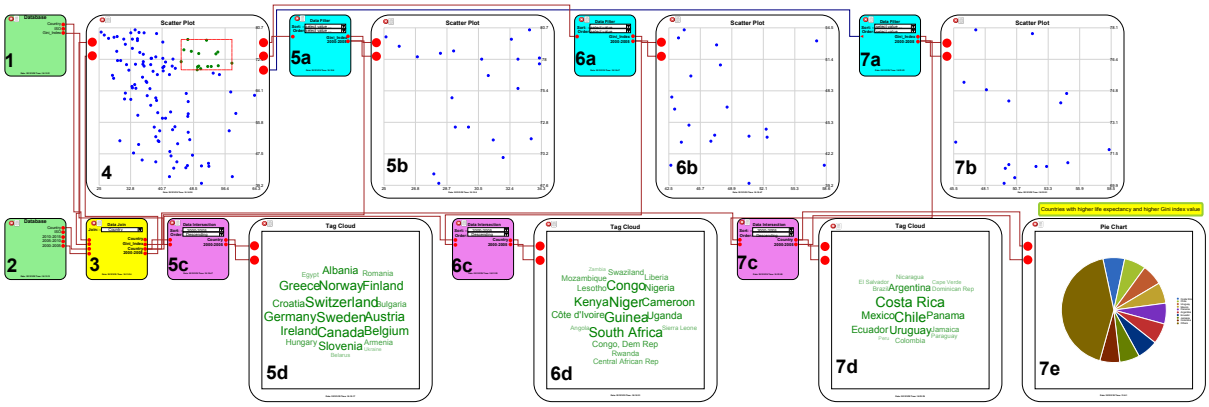

Figure 3: *The* ExPlatesJS *visual exploration system used to explore the world life expectancy at birth database and the world income equality Gini index database. The system allows for building visualization workflows using components such as datasets (green panels), inner joins (yellow), intersections (purple), and filters (cyan), interspersed with visualization panels.*

- **Database:** The database plate fetches the contents of an online data table using its URL. The system creates a unique output data anchor for each column extracted from the input data, labeling the anchor with the column name. Multiple data repositories can be loaded simply by creating multiple database plates, one for each repository.
- **Filter:** This plate supports filter its input data attributes based on user-defined parameters. Filter plates are automatically generated as a result of the user performing a filter operation inside a visualization plate.
- **Sorting:** Sorting plates are used to sort the rows of the input data dimensions in ascending or descending order with reference to a particular dimension.
- **Intersection:** This plate is used to perform an intersection operation on the indices of its input data rows. In other words, row indices that are common to **all** of its input anchors are passed on as output.
- **Union:** Union plates are used to perform a union operation on the indices of the input data.
- **Join:** Using the join plate, the user can perform an *inner join* operation on multiple data attributes from two or more different database plates. In order to perform this operation, user selects a common dimension as join key.
- **Frequency Counter:** The frequency counter is used to count the frequency of occurrence of unique values in the input. This plate takes a single input dimension and generates two outputs in response, one holding a list of unique values, and the other their corresponding frequencies.

5.6. Visualization Plates

Figure 2 shows an illustration of a visualization plate, where the visualization itself is shown in a viewport inside the plate. Since there exists no optimal visualization for visualizing complex multidimensional data [JE12], the user often needs to use different visualization techniques for different types of data, or even at different parts of the visual exploration process. We provide seven types of visualization plates in the current ExPlatesJS implementation.

- **Scatterplot:** A simple 2D scatterplot visualization supporting 2D filtering operation by drawing a bounding box.
- **Line Graph:** Renders multiple 1D input attributes as line graphs while supporting 1D filtering using a mouse drag.
- **Pie Chart:** An interactive pie chart taking two data wires as its input: the categories and the corresponding weights.
- **Bar Chart:** An interactive bar chart visualization accepting the categories (labels) and their respective values.
- **Tag Cloud:** Tag cloud of a frequency list of words; accepts one input for labels, another for frequencies.
- **World Map:** Thematic data visualized data on an interactive world map; two inputs provide latitude and longitude for data points, while any other input generates labels.
- **U.S. Map:** Similar to the world map, this plate allows for visualizing thematic data for USA. The user can filter the data by selecting different states using a mouse click.

5.7. Annotation

The ability to annotate and add notes to a visualization or visual exploration is key to promoting recall, reflection, and communication [TC05]. All plates in the system can be relabeled with a new title, and also have a comment field for descriptive text. There is also a separate annotation plate similar to a post-it note that the user can add text to and attach to any part of the canvas. Finally, ExPlatesJS also supports a free-form annotation mode that can be used to add handwriting to any part of the canvas (Figure 1).

6. Usage Scenarios

We here discuss two example scenarios of using ExplatesJS to explore two different types of real world datasets.

6.1. Scenario 1: World Demographic Data

In this first usage scenario, we explore two different data repositories for world demographic data simultaneously. We are interested in finding any relational trend between the two.

Dataset: The first data repository contains the demographic data of life expectancy at birth for 196 countries. The data values in this repository model the number of years a person is expected to live based on the country of birth. The second data repository is the Gini index values for 120 countries, specifying the income equality among the population of a particular country. The Gini index values vary from 0 to 100, where a higher value indicates a higher income inequality. Both of these data repositories are available on the World Resource Institute website (`http://earthtrends.wri.org/text/`) in CSV format.

Exploration: Figure 3 shows an example visual exploration for the two data repositories using ExPlatesJS. At the start, we imported the two repositories using their URLs into the system. This generated two database plates marked as **1** and **2** in Figure 3. Next, we performed a join operation on the selected attributes of the two data repositories using a join plate (**3**). In particular, we selected the country attribute, that is common in both the repositories, as the join key for the Gini index and the 2000-2005 life expectancy attributes from the two repositories. It is important to note that the two data repositories do not have identical country attribute values: the life expectancy data has 196 unique countries while the Gini index data has only 120 countries or rows. However, the output of the join yields only those data values for which a particular country is present in both repositories.

Findings: After joining the two attributes, we visualized them using a scatterplot visualization plate (**4**), taking the Gini index attribute on the horizontal axis and the life expectancy attribute on the vertical. The visualization shows that generally the countries having a higher Gini index value have lower life expectancy at birth. Interested in the countries having the best of these two attributes, we performed a filter operation by selecting the data points in the top left corner of the scatterplot. As a result, the system generated a filter plate (**5a**), containing the filtered data, and a scatterplot plate (**5b**), visualizing the filtered data. To find out which countries are included in this subset of the input data, we created an intersection data plate (**5c**) to intersect the data values at the output of the filter (**5a**) with all the countries at the output of the join plate (**3**). Next, we fed the output of this interaction operation to a tag cloud plate (**5d**) and noticed that countries like Switzerland, Sweden, and Canada are prominent in the selected data. We then performed a similar exploration (**6a, 6b, 6c, 6d**) for the countries having the worst of the two attributes. From this, we noticed that the African countries such as Guinea, Niger, and South Africa are prominent here, having a low life expectancy at birth and a high income inequality among their people. From the initial scatterplot visualization (**4**), we also observed that there are few data points, near the top right corner of the scatterplot visualization (**4**), that differ from the prominent trend between the life expectancy and the Gini index data values. On filtering and exploring these data points further (**7a, 7b, 7c, 7d, 7e**), we observed that these data points mainly correspond to the central and south American countries such as Costa Rica, Chile, and Argentina. In other words, although these countries have a higher income inequality in their population, people still have a reasonably high life expectancy.

6.2. Scenario 2: Worldwide Earthquake Data

Here we perform an ExPlatesJS visual exploration for the USGS earthquake (`http://earthquake.usgs.gov/earthquakes/catalogs/eqs1day-M2.5.xml`) live RSS feed and collection of Tweets.

Dataset: The United States Geological Survey (USGS), among other activities, monitors worldwide earthquakes and publishes several live RSS feeds of the latest earthquake data for different periods of time. In this scenario, we study worldwide earthquakes for the last seven days. The RSS feed consists of 10 different dimensions; for this example we focused on the five attributes magnitude, depth, latitude, longitude, and region. We also explore the live tweet data related to one of USGS Twitter accounts *@USGSBigQuakes*.

Exploration:

To explore this data (Figure 4), we first created a scatterplot visualization (**1**), taking the depth attribute on the horizontal axis, and the magnitude attribute on the vertical axis. Then, using a world map visualization plate (**2**) and our lat-lon attributes, we visualized the geographical locations (red circles on the map) of all these earthquakes. Next, with the help of frequency counter and pie chart plates, we highlighted the seismically active regions. We then performed three filter operations on the initial scatterplot plate, selecting magnitude ranges 2.5 to 4.0, 4.0 to 5.0, and 5.0 to 7.0, respectively. Each of these filter operations generated a filter plate and a scatterplot visualization plate (**4, 7, 10**), showing the filtered data. Next, we used the outputs of each of the filter plates as inputs to three intersection plates for the lat-lon and region attributes values. We feed the outputs of these intersections to three different world maps (**5, 8, 11**) and pie charts (**6, 9, 12**) to visualize the geographical locations of the filtered magnitude ranges and active regions prominent for these ranges. Finally, we used map and tag cloud plates (**13, 14**) to visualize tweets and their locations.

Findings: From the initial scatterplot visualization (**1**), we observed that in the past 7 days, there occurred many earthquakes of magnitude below 5.0 and a depth less than 130 km. We also observed that there were frequent earthquakes of magnitude above 5.0. From the geographical world map plate and pie chart plate at the top (**2, 3**), we noted that the African continent had been most stable in the last 7 days, while Alaska, the Dominican Republic, and Japan were most

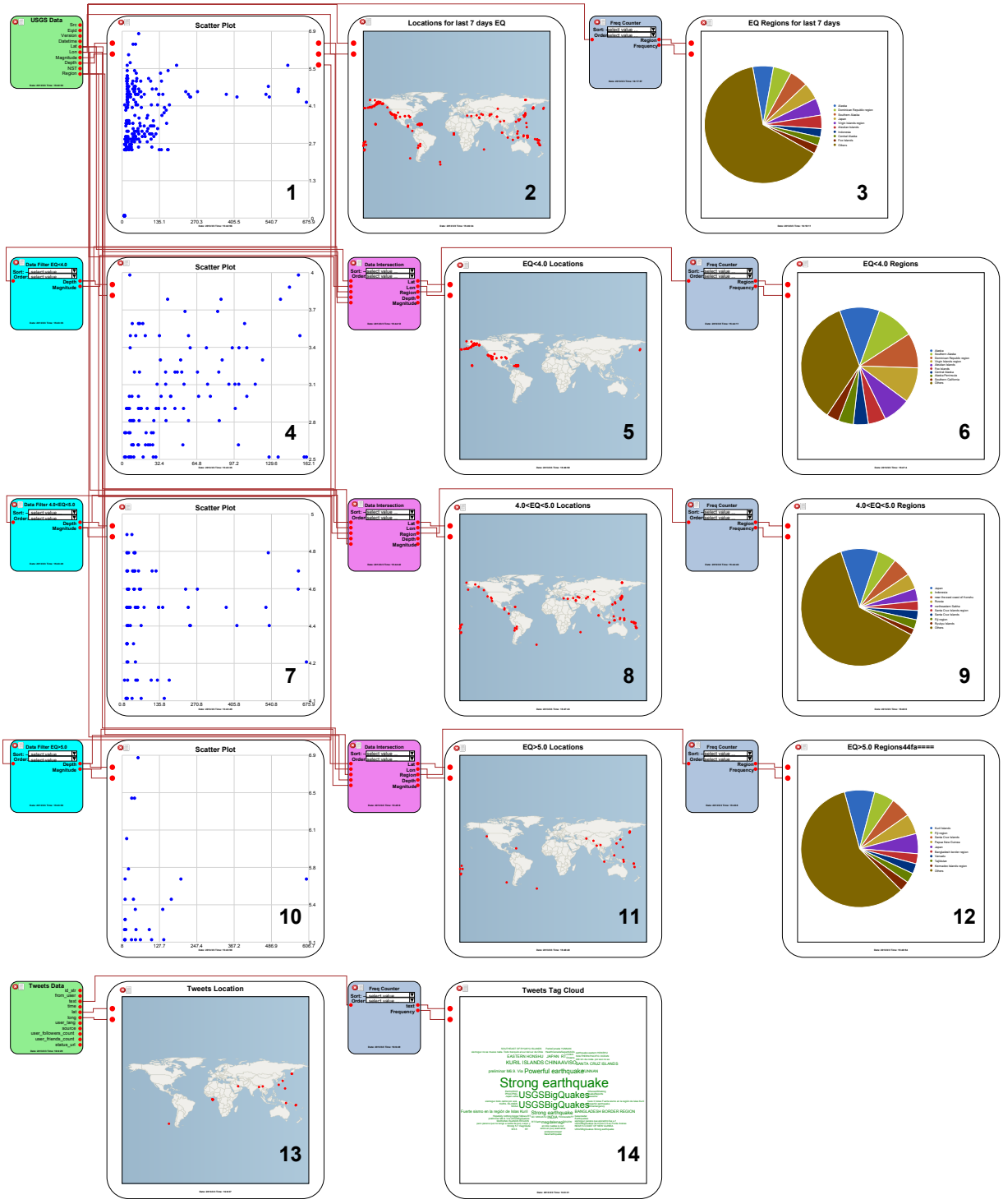

Figure 4: *USGS earthquake data scenario. The live RSS feed (i.e., it can be refreshed to receive new data) shows the last seven days of significant earthquakes. The user is filtering by magnitude to see where different scale earthquakes are concentrated.*

prominent among the seismically active regions. From the exploration of different magnitude ranges, we came across an interesting observation: the majority of the low magnitude earthquakes, below 4.0, occurred near or inside North America (**4, 5, 6**). The implication is that although this area was not very stable in the past 7 days, earthquakes that occurred here were of unnoticeable magnitude. Furthermore, we found that in the past 7 days, the earthquakes of magnitude between 4.0 and 5.0 were generally occurring in the South American and Asian regions (**7, 8, 9**). Finally, we also observed that the majority of the earthquakes above 5.0 occurred in the far east Asia region, indicating the high instability of this region (**10, 11, 12**). From the tweets plates (**13, 14**), we observed that not only the regions of high magnitude earthquakes are prominent among these tweets, most of these tweets also originated from the nearby regions.

7. Discussion

Scalability: Probably the most important challenge faced by a system like ExPlatesJS is its scalability in the presence of multiple exploration states, giving rise to issues on (a) navigation, (b) visual clutter, and (c) usability. To overcome the navigation issue, ExPlates provides zooming and panning, supported by the visual canvas, and data flow awareness, achieved through data wires. It is possible for the user to zoom to a particular exploration state and interact with it without any interference from other states, essentially making it a classic visualization system. However, an excess of data wire across the visual canvas comes at the cost of additional visual clutter. The ExPlatesJS system tries to minimize this visual clutter by placing wires in channels of empty space between the plates. Furthermore, in order to identify a specific data wire, the user can hover the mouse on its associated data anchors, highlighting the associated wire by temporarily changing its color and increasing its thickness.

Finally, ExplatesJS is a web-based system, and scalability is a challenge for any web-based visualization systems. We have tested ExPlatesJS with datasets containing tens of thousands of items, and still achieve interactive performance.

Complexity: There is a often clear tradeoff between the complexity of a system and the benefits achieved through it. Even if ExPlatesJS is accessible as a web application, it is clearly not intended for novice users. At the same time, the individual plates in ExPlatesJS have been designed to be simple building blocks. We believe that if a novice user can only understand the basic conceptual model of the data flow in ExPlates, they should also be able to compose complex queries from simple visualization and analytics components.

Comparison: Multidimensional data visualization and analysis is nowadays a commercial venture, with tools such as Spotfire [Ahl96], Tableau (formerly Polaris [SH00]) being key players. The main difference between these tools and ExPlatesJS is that we use a data flow approach as opposed to the workbench or dashboard common to traditional spreadsheets or statistics packages. This approach makes it easier for us to externalize not just the current view, but also the sequence of views that lead to it. Having said that, other than Spotfire and Tableau being mature commercial software for desktop computers and thus having a wealth of more advanced features, there are no fundamental functional differences between them. Views and analyses from one can be replicated in the other; it is just our emphasis on the exploration process that is different.

Beyond general multidimensional tools, ExPlates can also be compared to work on graphical histories, the most relevant being a comic strip history interface [HMSA08]. Just like the ExPlatesJS system, this approach helps the user to overview and recall steps performed through the course of an exploration. However, in ExPlatesJS, the graphical history is an integral part of the exploration interface, whereas Heer's work instead uses a sequence of thumbnails. Their approach is more space-efficient, but our approach allows for comparing exploration states side-by-side as well as better understanding the data flow and relation between states.

Data flow systems are another valid comparison, where the most relevant example is the GraphTrail [DRL*12] system for large multivariate graphs. Although GraphTrail is similar to ExPlatesJS, albeit for networks, there are several major differences. In particular, GraphTrail views are more monolithic and heavyweight objects, and are not as extensible as the SVG-based objects of ExPlatesJS, which supports a more relaxed architecture by differentiating between data and visual transformations independently. This is particularly important for a diverse data environment like the web, where data can be retrieved from different data repositories and it is not always possible to feed the extracted data directly to a visualization, as highlighted in Scenario 1.

8. Conclusion and Future Work

We have presented a new approach to capturing and externalizing visual exploration by *spatializing* all mutating operations conducted on a visual representation using an infinitely zoomable and pannable exploration canvas. The intuition is that instead of changing an existing visualization in-place, we spawn a new immutable visualization where the desired change has been performed. Several possibilities exist for future work in this domain. Overall, we think that the spatialization model holds considerable promise, and we look forward to designing additional visualization systems and techniques based on this approach.

Acknowledgments

This work was partially supported by the U.S. National Science Foundation under grant TUES-1123108. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[Ahl96] AHLBERG C.: Spotfire: an information exploration environment. *SIGMOD Record 25*, 4 (1996), 25–29. 9

[APM*11] ANDERSON E. W., POTTER K. C., MATZEN L. E., SHEPHERD J. F., PRESTON G., SILVA C. T.: A user study of visualization effectiveness using EEG and cognitive load. *Computer Graphics Forum 30*, 3 (2011), 791–800. 2

[BCS*05] BAVOIL L., CALLAHAN S. P., SCHEIDEGGER C. E., VO H. T., CROSSNO P., SILVA C. T., FREIRE J.: VisTrails: Enabling interactive multiple-view visualizations. In *Proceedings of the IEEE Conference on Visualization* (2005), pp. 135–142. 4

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 6 (2011), 2301–2309. 5

[BPW*93] BRODLIE K., POON A., WRIGHT H., BRANKIN L., BANECKI G., GAY A.: GRASPARC: A problem solving environment integrating computation and visualization. In *Proceedings of the IEEE Conference on Visualization* (1993), pp. 102–109. 3, 5

[Cle94] CLEVELAND W. S.: *Visualizing Data*. Hobart Press, 1994. 1

[CMS99] CARD S. K., MACKINLAY J. D., SHNEIDERMAN B. (Eds.): *Readings in Information Visualization — Using Vision to Think*. Morgan Kaufmann, 1999. 2

[DR01] DERTHICK M., ROTH S. F.: Enhancing data exploration with a branching history of user operations. *Knowledge-Based Systems 14*, 1-2 (2001), 65–74. 3, 5

[DRL*12] DUNNE C., RICHE N. H., LEE B., METOYER R., ROBERTSON G.: GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the ACM Conference on Human Factors in Computer Systems* (2012), pp. 1663–1672. 4, 9

[EST07] ELMQVIST N., STASKO J., TSIGAS P.: DataMeadow: a visual canvas for analysis of large-scale multivariate data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology* (2007), pp. 187–194. 4

[GS06] GROTH D. P., STREEFKERK K.: Provenance and annotation for visual exploration systems. *IEEE Transactions Vis. Comput. Graph 12*, 6 (2006), 1500–1510. 4

[GZA06] GOTZ D., ZHOU M. X., AGGARWAL V.: Interactive visual synthesis of analytic knowledge. In *Proceedings of IEEE Symposium on Visual Analytics Science & Technology* (2006), pp. 51–58. 2, 3

[HMSA08] HEER J., MACKINLAY J. D., STOLTE C., AGRAWALA M.: Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1189–1196. 2, 3, 9

[HS12] HEER J., SHNEIDERMAN B.: Interactive dynamics for visual analysis. *Communications of the ACM 55*, 4 (2012), 45–54. 3

[JE12] JAVED W., ELMQVIST N.: Exploring the design space of composite visualization. In *Proceedings of the IEEE Pacific Symposium on Visualization* (2012), pp. 1–8. 4, 6

[JKMG02] JANKUN-KELLY T. J., MA K.-L., GERTZ M.: A model for the visualization exploration process. In *Proceedings of the IEEE Conference onVisualization* (2002), pp. 323–330. 2, 3

[Kei02] KEIM D. A.: Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 1–8. 1, 2

[KF88] KURLANDER D., FEINER S.: Editable graphical histories. In *Proc. of IEEE Workshop on Visual Language* (1988), pp. 127–134. 2, 3

[LS87] LARKIN J., SIMON H.: Why a diagram is (sometimes) worth 10000 words. *Cognitive Science 11* (1987), 65–69. 3

[Ma99] MA K.-L.: Image graphs - A novel approach to visual data exploration. In *Proceedings of the IEEE Conference on Visualization* (1999), pp. 81–88. 3

[McK09] MCKEON M.: Harnessing the information ecosystem with wiki-based visualization dashboards. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1081–1088. 4

[Nor93] NORMAN D. A.: Cognition in the head and in the world: An introduction to the special issue on situated action. *Cognitive Science 17*, 1 (1993), 1–6. 3

[PSM07] POUSMAN Z., STASKO J. T., MATEAS M.: Casual information visualization: Depictions of data in everyday life. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1145–1152. 4

[RSPC93] RUSSELL D. M., STEFIK M. J., PIROLLI P., CARD S. K.: The cost structure of sensemaking. pp. 269–276. 2

[SH00] STOLTE C., HANRAHAN P.: Polaris: a system for query, analysis and visualization of multi-dimensional relational databases. In *Proceedings of the IEEE Symposium on Information Visualization* (2000), pp. 5–14. 9

[SH10] SEGEL E., HEER J.: Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1139–1148. 3

[Shn96] SHNEIDERMAN B.: The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages* (1996), pp. 336–343. 2

[SR96] SCAIFE M., ROGERS Y.: External cognition: how do graphical representations work? *International Journal of Human-Computer Studies 45*, 2 (1996), 185–213. 2, 3

[SvW08] SHRINIVASAN Y., VAN WIJK J.: Supporting the analytical reasoning process in information visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2008), pp. 1237–1246. 3, 5

[TC05] THOMAS J. J., COOK K. A. (Eds.): *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005. 2, 3, 6

[TFF88] TUKEY J. W., FISHERKELLER M. A., FRIEDMAN J. H.: PRIM-9: An interactive multi-dimensional data display and analysis system. In *Dynamic Graphics for Statistics*. Wadsworth & Brooks/Cole, 1988, pp. 111–120. 1

[Tuk77] TUKEY J. W.: *Exploratory data analysis*. Addison-Wesley, 1977. 1

[WSP*06] WRIGHT W., SCHROH D., PROULX P., SKABURSKIS A., CORT B.: The sandbox for analysis: Concepts and evaluation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2006), pp. 801–810. 2, 4

[YaKSJ07] YI J. S., AH KANG Y., STASKO J. T., JACKO J. A.: Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1224–1231. 2