Using Animation in Multi-Class Scatterplot Matrices

Helen Chen, Sophie Engle, Alark Joshi, Eric D. Ragan, Beste F. Yuksel, and Lane Harrison

Abstract—The scatterplot matrix is one of the most commonly used techniques for visualizing multi-variate data. While scatterplot matrices work well to show correlations and distributions of data for small datasets, they suffer from overdraw (overlapping points) when used for visualizing medium to large datasets. There are many approaches focused on alleviating overdraw in single-class scatterplots, but far fewer approaches focus on the harder case of multi-class scatterplot matrices. We introduce a novel yet simple approach using animation that alleviates overdraw for these multi-class scatterplot matrices. By continuously drawing the underlying data, this animated approach enables users to discriminate between regions of varying density (number of points) and diversity (number of classes). In a user study with 69 participants, we found that animation not only improved the ability of users to identify dense regions, but also that users found animated SPLOMs easier to interpret and preferred them over static SPLOMs. We can use this animated approach on its own or to augment existing techniques, making it a powerful addition for alleviating scatterplot overdraw.

Index Terms—Animation, overdraw, multi-class scatterplot matrix

1 Introduction

Scatterplot matrices (SPLOMs) are one of the most widely used techniques for visualizing multi-variate data [13]. Through a matrix of scatterplots, SPLOMs enable viewers to analyze both an overview of the pair-wise relationships between variables in the dataset, as well as relationships within an individual plot. Because SPLOMs use spatial position as their primary encoding, they are effective tools for identifying correlations, clusters, outliers, and other features of interest in multi-variate data [52].

Fundamental challenges remain, however, when using color to represent multi-class data in SPLOMs. Consider the widely known "diamonds" dataset [56], which contains not only continuous variables, but also categorical variables that turn out to be critical for uncovering underlying patterns in the data. Visualizing this dataset in a multi-class SPLOM results in significant overdraw. *Overdraw* occurs when points or glyphs are drawn on top of each other and occlude the underlying data. For moderate to large datasets like this one, overdraw affects the ability of viewers to accurately understand the data distribution and density, as well as discern relationships amongst subgroups of the data (see Section 3).

Common approaches to alleviate overdraw are not sufficient for this case, as seen in Figure 1. Jittering, for example, relies on datasets having low density. Similarly, opacity modulation can scale to higher densities, but interferes severely with color perception [52]. Traditional hex binning only works for single-class scenarios [9]. New, broadly applicable approaches are needed to support exploratory data analysis with multi-variate multi-class data.

In this paper, we propose and evaluate a straightforward animation technique for alleviating overdraw in multi-class SPLOMs. This technique aims to fulfill Munzner's requirement to provide points with "guaranteed visibility over time" [40]. The technique involves continuously looping through the underlying dataset and redrawing one or more rows of points in each animation frame. This effect causes flickering to occur in highly overdrawn regions, particularly those with high density and high numbers of classes. Importantly, the technique does not modify the original visual encodings, preserving the appearance of features such as outliers. As we discuss in Section 2.2, prior research has already established that the type of flickering produced by this technique can draw user attention while remaining unobtrusive when the user needs to attend to specific data features.

We demonstrate in Section 5 how our interactive tool may be used to explore the canonical multi-class "diamonds" dataset [56]. Despite needing to render over a million individual points across several buffers, we were able to successfully use our web-based tool to identify regions of varying density and diversity in this complex dataset.

We also report the results of a user study with 69 participants who performed tasks using both animated and static SPLOMs to determine the impact animation had on alleviating overdraw. Participants were not only able to identify denser regions in the animated version, but were also able to complete these tasks in comparable time to the static version. Most participants also indicated that animated SPLOMs were easier to interpret than static and preferred the animated approach. However, some participants found animation distracting and preferred static SPLOMs instead.

These results suggest that animation is a promising approach for alleviating overdraw in multi-class scatterplot matrices. More broadly, they contribute to the ongoing discussion of the role of animation in visualization. In this case, a simple application of animation provides tangible benefits for exploratory data analysis.

Thanks to the simplicity of this novel approach, we are also able to provide an interactive web-based proof of concept capable of handling datasets with tens of thousands of rows. We encourage readers to visit vgl.cs.usfca.edu/animated-sploms to interact with our tool.

2 Background

We next discuss background and related work on scatterplots, scatterplot matrices, and traditional uses of motion and animation in visualization.

2.1 Scatterplots and Scatterplot Matrices

Scatterplots are a common technique to visualize bivariate data with variations to visualize more variables by modifying the shape, color, and size of the plotted points [23]. Our work focuses on the scatterplot matrix (SPLOM) technique that is used to visualize multivariate datasets by placing pairwise scatterplots of the variables in a matrix.

Scatterplots have been actively researched in the information visualization community. For example, Doherty et al. conducted a user study on estimating correlations using scatterplots and found that the overall perception that individuals underestimate correlations is inaccurate [18]. The study found that participants were fairly accurate with low correlations, overestimated midrange correlations, and underestimated high correlations. Bachthaler and Weiskopf extended the power of scatterplots to visualize continuous data frequently generated in computational simulations [1].

Gleicher et al. performed a large scale user study that evaluated participants' ability to compare the means of multiple classes being

- *All non-first authors are ordered by date they joined the project.*
- *Helen Chen, Sophie Engle, Alark Joshi, and Beste F. Yuksel are with the University of San Francisco. E-mail: [hhchen, sjengle, apjoshi, byuksel]@usfca.edu.*
- *Eric Ragan is with Texas A&M University. E-Mail: eragan@tamu.edu.*
- *Lane Harrison is with Worcester Polytechnic Institute. E-Mail: lane@cs.wpi.edu.*

displayed [24]. They found that using color to distinguish multiple classes is better than shape and combining shapes redundantly does not improve performance. Additionally, encoding data with redundant cues for tasks involving individual classes did not affect user performance.

Lehmann et al. present an approach to explore large scatterplot matrices (SPLOMs) by quickly selecting "relevant" plots [35]. Their system helps users interactively find similar/different scatterplots in a scatterplot matrix to explore the data. They use an image-based metric to identify similar/completely different scatterplot cells in a scatterplot matrix.

2.2 Motion and Animation

Motion has been shown to be highly effective at drawing attention [45] and is also easily perceived in the peripheral vision [41]. Motion is a known preattentive attribute [29], and is often broken into three categories: flicker, direction of motion, and velocity of motion [28]. Our animated approach uses flickering—points disappear and reappear over time, but they do not move in position. Huber and Healy demonstrated that flicker must have a cycle length of 120 milliseconds or greater for accurate detection [30]. We animate at a rate of 30 frames per second, but the exact rate that points flicker depends on the density in that region. As a result, it is possible that points flicker faster than viewers are able to identify individual points. However, our approach uses flickering to draw attention to dense regions, not to identify individual points. This is an important distinction in how we use animation. Existing work often uses motion to focus attention on a specific element, whereas we use it to focus attention to regions composed of multiple elements.

The related work by Bartram et al. found that anchored motions like flickering were less distracting than other types of motion [2]. This helped alleviate concerns that animating the entire scatterplot matrix at once would be too distracting for users. Indeed, as we discuss in Section 6.3, we confirmed that most users still performed well (and preferred) the animated approach even though some found animation somewhat distracting.

Animation has been a topic of active research in the field of data visualization with several studies finding situations in which animation was helpful (e.g., [4, 17, 31, 43]) or detrimental to the task at hand (e.g., [3, 44]). For example, Heer and Robertson found that animated transitions between different representations significantly improved graphical perception [31], but Robertson et al. later found that animation of trend traces was less effective compared to static alternatives [44].

Animation and movement are also frequently used in visualization to show changes over time. For example, Tversky et al. identified two guiding principles when animation may be successful in conveying changes over time [49]. Craig et al. present an interactive technique that allows biologists to explore large time-varying microarray data [15]. They used animation to convey changes in the number of genes across the selected frames through various scatterplots that are plotted at individual instants in the time frame. Griffin et al. also explored animated versus small-multiple maps for time-varying data, and they found animation produced better results for detecting clusters moving over time [25]. Our approach differs in its use of animation from these examples in that we do not use it to show changes over time.

Animation and motion have been used for other reasons as well. Ware and Bobrow examined the use of oscillatory motion to highlight subsets of a node-link diagram [52]. Ellis and Dix suggest animation as a strategy to reduce clutter in data visualization since it addresses problems associated with overdraw such as occlusion of data items [20]. Shearer et al. present the use of animation in different types of data and visualization scenarios: geographic data, hierarchical data using a treemap, multi-attribute data using parallel coordinates, and a graph using node-link diagram [48]. The motivation to use animation is that continuous sampling could quickly provide the user with an overall sense of the data in an overcrowded display. While our work is closely related, they did not evaluate their technique for the specific case of multi-class scatterplots.

Animation has also been used to convey uncertainty. For example, the work by Feng et al. uses kernel density estimation to define a probability distribution function that is used to communicate uncertainty in parallel coordinates and scatterplots [22]. Although the technique is excellent at conveying uncertainty, the ability to discriminate individual points is lost due to their approach to display density-based plots. They use a random sampling technique to animate probabilistic plots, which results in flickering outliers. The disappearing outliers can pose a problem since the viewer does not know when or where they are going to show up. This effect is even more pronounced in a scatterplot matrix, where a user may be examining a specific scatterplot and could miss an outlier flicker in a different part of the matrix. Our technique does not rely on sampling and takes the opposite approach—outliers appear stable and dense regions flicker.

3 Overdraw

One of the major problems with visualizing scatterplots and scatterplot matrices is with overdraw for medium to large datasets. Overdrawing (or overplotting) refers to multiple data elements being resolved to the same pixel location on the visualization. A moderate to large amount of overdraw hinders the viewer's ability to discern any meaningful information from the visualization.

3.1 Simple Approaches

Approaches for alleviating overdraw include using different types of glyphs (shapes [34], sunflowers [14]), varying the sizes of glyphs depending on the amount of overplotting [36], applying alpha blending [12], and applying a small amount of noise to the dataset (e.g. jittering [10]), but these approaches are less effective when applied to larger and denser datasets. Carr et al. [9] developed approaches to address problems associated with visualizing dense scatterplots "with large N". They introduced the use of drawing contours around dense regions, hexagonal binning, and animation as alternative techniques to deal with large, dense scatterplots. They also proposed animating subset sequences (or classes) with respect to interactivity.

Dang et al. addressed the general challenge of overdraw in data visualization by introducing stacking in three-dimensional space [16]. Sedlmair et al. [47] later examined the ability of various dimensionality reduction techniques [50] (such as PCA [32] and MDS [7]) to separate classes when visualized using 2D scatterplots, scatterplot matrices, and 3D scatterplots. They found that users performed sufficiently well for class separability tasks when using 2D scatterplots and using a scatterplot matrix was valuable at times, but 3D scatterplots were found to hurt a user's ability to discriminate between various classes due to visual clutter.

3.2 Advanced Approaches

Bertini and Santucci developed a framework to measure the "degradation" in quality when visualizing large datasets using a scatterplot [5]. Additionally, they developed an automatic sampling strategy that addresses the degradation in quality that a viewer experiences. In a subsequent paper, Bertini and Santucci further fine-tuned their sampling strategy to include best uniform sampling and non-uniform sampling to reduce the observed degradation in quality [6]. Their collaborators were able to visualize sub-groupings as well as identify subtle patterns in their time-varying data.

Keim et al. presented the *generalized scatterplots* technique which uses pixel-based techniques to address overdraw [33]. The proposed techniques allow users to vary the acceptable degree of overlap and "place identical data points in a neighborhood around the already plotted pixels." Hao et al. extended the idea to use aggregation in regions with overdraw and provide the user with a binned representation of the data [26].

Eisemann et al. presented an interactive technique to explore dense scatterplots [19]. Their technique allows users to add a semantic lens in regions of overdraw to explore the region in more detail. This technique requires fine-tuning the axes of the lens and may result in loss of context due to the occlusion of underlying data.

Matejka et al. identified a novel opacity-optimization technique to control the visual clutter observed due to overdraw in scatterplots [37]. They allowed users to select opacity levels when viewing scatterplots

of large datasets. Based on their crowd-sourced model, they scale the opacity of the displayed points during overdraw in scatterplots.

To address the overdraw in scatterplot matrices, Elmqvist et al. developed a technique that uses the scatterplot matrix as an overview as well as a navigational interface [21]. They allow users to transition from one scatterplot to another to compare and correlate the attributes in their data.

All of these techniques are alternatives for alleviating overdraw in *single-class* scatterplots or scatterplot matrices. However, these techniques are limited in their applicability to *multi-class* settings. As we discussed in the introduction, our approach is targeted towards alleviating overdraw in multi-class scatterplot matrices.

3.3 Multi-Class Approaches

There has also been some work exploring how to address overdraw for multi-class scatterplots. For example, splatterplots can use a contour-based technique to address overdraw, and Mayorga and Gleicher showed this approach can work for multi-class settings [38]. They calculated contour-bounded filled areas in dense regions and performed subsampling outside those regions to preserve outliers. However, increasing the number of classes or subgroups increases the visual complexity due to the interaction between overlapping subgroups.

Chen et al. also addressed overdraw in multi-class scatterplots by using a multi-class blue noise sampling scheme to estimate point density [11]. Their sampling scheme eliminates occlusion and produces consistent scatterplots regardless of the drawing order.

These two techniques represent the state-of-the-art when it comes to handling overdraw issues in multi-class scatterplots, but both rely on sampling and do not address overdraw in scatterplot matrices. In fact, we were not able to identify *any* literature addressing overdraw in multi-class scatterplot matrices. Our animated approach is designed specifically to apply to multi-class scatterplot matrices, and could be used to augment these and other sampling-based approaches to alleviating overdraw. We discuss this as a potential direction of future study in Section 7.

4 Approach

The idea behind our approach is to continuously redraw points by looping over the dataset, allowing for overdrawn points to eventually reappear. See Figure 2 for an illustration of our approach and tool, or see vgl.cs.usfca.edu/animated-sploms for a live demo.

4.1 Environment

Knowing that we wanted an interactive web-based tool, we implemented a prototype of our tool using D3.js [8, 55] (version 3) and SVG elements. To animate the SPLOM, we looped through the dataset row by row using a timer. For each circle associated with that row in the SVG, we removed and re-inserted that circle so it would be drawn on top. We quickly ran into performance issues animating moderate sized datasets with this approach.

We decided to switch to using the HTML5 canvas and the P5.js [60] JavaScript library, since our animation paradigm fit naturally with the P5 draw loop. We were able to animate the larger datasets with fewer performance issues, but had to rethink how to implement interactivity in our tool. We detail our P5 implementation in the following sections.

4.2 Scatterplot Matrix

We start with a traditional scatterplot matrix. Since the cells along the diagonal and lower triangle are redundant, we only draw cells in the upper triangle and rotate the triangle so that it lies in the upper-left quadrant. Each cell is fixed in size, so the overall size of the SPLOM depends on the number of variables being visualized. For each row and column, we draw the variable name and range (minimum, middle, and maximum values). We place a legend and other controls in unoccupied cells in the matrix, as depicted in Figure 2c–2e.

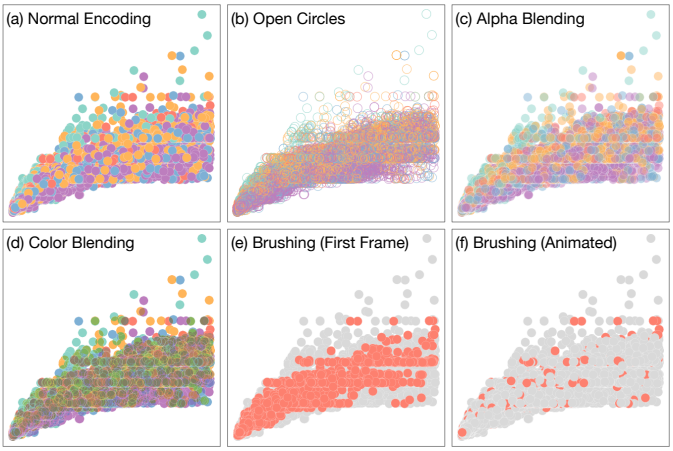

Fig. 1. The *carat* versus *price* cell from the *diamonds* SPLOM using our tool. **(a)** The default point encoding, which uses opaque circles filled by class color and a thin white stroke. **(b)** The open circle encoding, which uses a thick stroke colored by class and no fill. **(c)** The alpha blending encoding, which uses slightly transparent fill colors. **(d)** The color blending encoding, which uses the "DARKEST" color blending mode in P5. **(e)** The first frame from brushing the red *Good* class using normal encoding, with animation paused. **(f)** Following frames after brushing the red *Good* class and re-enabling animation.

4.3 Point Encoding

We adapted a qualitative color scheme from ColorBrewer [27, 54] to encode the class of each point. By default, the points have a fully opaque (not transparent) fill based on their class with a thin white stroke. See Figure 1a for an example. The white stroke ensures that points of the same color are still distinguishable from each other.

We also implemented other point encodings that are commonly used to alleviate overdraw. Open circles, depicted in Figure 1b, uses a thick stroke colored by class and no fill for each point. Alpha blending, depicted in Figure 1c, is the same as our default encoding except the fill and stroke of each point are slightly transparent. Finally, Figure 1d depicts the "DARKEST" mode of color blending used in P5.

However, using these encodings stand-alone with more than four colors can give mixed results for diverse regions with significant overdraw. For example, consider the static snapshots in Figure 1. It is difficult to determine the most dense and diverse regions (it all appears dense and diverse), as well as determine any relationships between the classes. We will show in Section 5 that animation paired with interaction enables those kind of conclusions.

4.4 Animation

Our approach to animating the scatterplot matrix is to continuously loop through the dataset and redraw one or more rows of points at a time. For example, drawing one row for a dataset with five variables (not including class) results in drawing ten points: one for each pair of variables, or each cell in our matrix.

To provide context to the viewer, we do not wipe the canvas clean after each draw loop, thus keeping the entire dataset visible at all times. A progress bar, shown above the legend in Figure 2d, shows the percentage of rows drawn, and then the percentage of rows that have been re-animated for every subsequent loop through the data.

Using this technique, the amount of visible redraw or movement indicates the amount of overdraw in a region. Points that are overdrawn will reappear during later draw loops. Outliers are perceived to be static, even when redrawn. Similarly, regions with low density (i.e., few points) and low diversity (i.e., few classes) will have little to no movement and be perceived to be static. Regions with high density but low diversity will have movement due to the white outline of each point, but will not change in color (since each class is represented by a color). Regions with both high density and high diversity will change color

frequently. This is the case in which animation successfully draws attention through flickering to these diverse and dense regions that could potentially be misinterpreted due to overdraw in static SPLOMs.

4.5 Parameters

Our tool supports several URL query string parameters to control how points are rendered. Specifically:

- **Animation Speed:** We draw one row of points per frame with a frame rate of 30 frames per second by default. To reduce the time it takes to animate the entire dataset, users can speed up the animation by increasing the number of rows drawn per frame using the `animateNum` parameter. For example, vgl.cs.usfca.edu/animated-sploms/?animateNum=5 will animate five rows per frame.
- **Pre-Rendering:** The entire dataset is not fully drawn until the first animation loop is completed, which can cause issues if outliers appear towards the end of large datasets. To ensure all outliers are immediately visible, our tool supports rendering the entire dataset prior to the first animation loop using the `initDraw` parameter. Most datasets have this parameter enabled by default. For example, vgl.cs.usfca.edu/animated-sploms/?initDraw=false turns off pre-rendering.
- **Point Encoding:** Users can change the point encoding to one of the other options discussed in Section 4.3 by setting the `encoding` parameter to `open`, `alpha_blended`, or `filled_blended`. For example, vgl.cs.usfca.edu/animated-sploms/?encoding=open will use circles with a thick stroke and no fill instead of the default encoding. See Figure 1b, 1c, and 1d for examples.
- **Scale Amount:** Each cell is fixed in size. Users may scale the size of the cells by setting the `scaleAmount` parameter. Values less than 1 will shrink the cell size, and values 1 or greater will increase the cell size. For example, the example shown at vgl.cs.usfca.edu/animated-sploms/?scaleAmount=1.5 increases the cell size by 1.5 times the original size.

The animation speed parameter can also be controlled via a spinner widget, as shown in Figure 2c. Users also have the option to pause or reactivate the animation using the play/pause buttons next to the spinner.

4.6 Interactivity

When users hover over a cell in the SPLOM, the associated row and column labels on the left and top of the matrix are highlighted. We also implemented brushing by class. This was slightly challenging after moving to P5 from D3, since we could no longer simply select the appropriate circles to highlight. Instead, we decided to use several off-screen buffers to implement brushing. This process is illustrated for our synthetic dataset in Figure 2.

Specifically, for every frame, we rendered each point being animated to three buffers: a buffer with all points colored, a buffer with all points in light grey, and a class-specific buffer. Consider the example in Figure 2. For example, if we were rendering a point belonging to the *A* class, we would draw it in the "All," "Grey," and "Class A" buffers. Since there are four classes in this example, we need six buffers total.

Users may select one or more classes by clicking on the corresponding color square(s) in the legend, as depicted in Figure 2e. When users select a class, the appropriate buffers are displayed. The example shows classes *B* and *C* brushed, which requires the "Grey," "Class B," and "Class C" buffers to be shown. Any combination of classes may be selected. When all classes are selected, only the "All" buffer is shown. When users first brush a class, points from non-brushed classes are shown in the background in grey and points from all brushed classes are shown in the foreground in their appropriate class color. For example, Figure 1e shows the first frame after brushing the red *Good* class. If animation is enabled, then points belonging to non-brushed classes may start to be redrawn on top of the other points, as depicted in Figure 1f. Users can brush the data regardless of whether or not the animation is paused.

5 Usage Scenario

We demonstrate the utility of our approach by visualizing the diamonds dataset [56] included in the ggplot2 package [53, 58] in R containing the prices and other attributes of round-cut diamonds. This dataset includes 53,940 rows, 3 categorical columns, and 7 continuous columns. We visualize the the *x*, *y*, *z*, *price*, *table*, *depth*, and *carat* columns in the SPLOM and use the *cut* column with 5 categories to color each point. Visit vgl.cs.usfca.edu/animated-sploms to view an animated live demo of this dataset and Figure 3 for an illustration of the regions highlighted.

5.1 Observations

The following observations were made on a 2013 iMac Desktop animating 30 rows per frame after the entire dataset was loaded. We first focused on whether we could identify outliers, and regions of varying density and diversity:

- Look at the *z* column on the left-hand side in the SPLOM. There is a single point of the blue *Very Good* class with a *z* value near 35 (see Figure 3a). The lack of movement indicates this is a region with few points, and the lack of flickering colors indicates this region has few classes. We confirmed in the dataset that there is only a single row with a *z* value of 31.8 and all other *z* values are under 10. This is an **outlier** in a **low density** region.
- We did not observe any clear **outliers** in **high density** regions, but it may be difficult to perceive those given the time it takes to animate the entire dataset. We were able to identify outliers in high density regions in synthetic datasets smaller in size.
- Look at the *depth* row second from the top in the SPLOM. There is a region with *depth* values greater than 65 of the teal *Fair* class (see Figure 3b). The lack of movement indicates there are few points. We confirmed that only 2% of all rows have *depth* values in this range. The lack of flickering colors indicates this region has few classes. We confirmed in the dataset that *Fair* points make up 93% of the points in this region. This is a **low density** and **low diversity** region.
- Look at the cell for *carat* and *table* in the upper-left of the SPLOM (see Figure 3c). There is a region of points with *carat* values above 2.6 and several classes, but with little movement. We confirmed that this region contains less than 0.2% of all the rows, but all classes. This is a **low density** and **high diversity** region.
- There are no regions with frequent movement and little flickering of colors. We can conclude that the densest regions in this dataset also tend to be diverse, or in other words, there are no regions with **high density** and **low diversity**.
- Look at the *depth* and *table* cell in the upper-left of the SPLOM (see Figure 3d). There is a region of points with both *depth* and *table* values between 55 and 65. There is a large amount of movement in this region indicating it contains many points. There is also a large amount of flickering colors indicating it contains many classes. We confirmed this region contains 80% of the points and all five classes. This is a **high density** and **high diversity** region.

We were also able to make other general observations using our tool, and confirmed those observations by looking at the raw data:

- The points for the teal *Fair* class are noticeable overall, since these points tend to occupy regions with low diversity. However, this class represents only 3% of the dataset. If we brush only the teal *Fair* class and focus on the *carat* versus *price* cell (see Figure 3e), it is more apparent that these points make up a small percentage of the dataset.
- There is a dense and diverse region of points in the *depth* versus *table* cell in the SPLOM with values between 55 and 65 (see Figure 3d). In the default view, it is difficult to determine if there are points of the teal *Fair* class in this region. This class appears infrequently in this region if we brush only this class. We

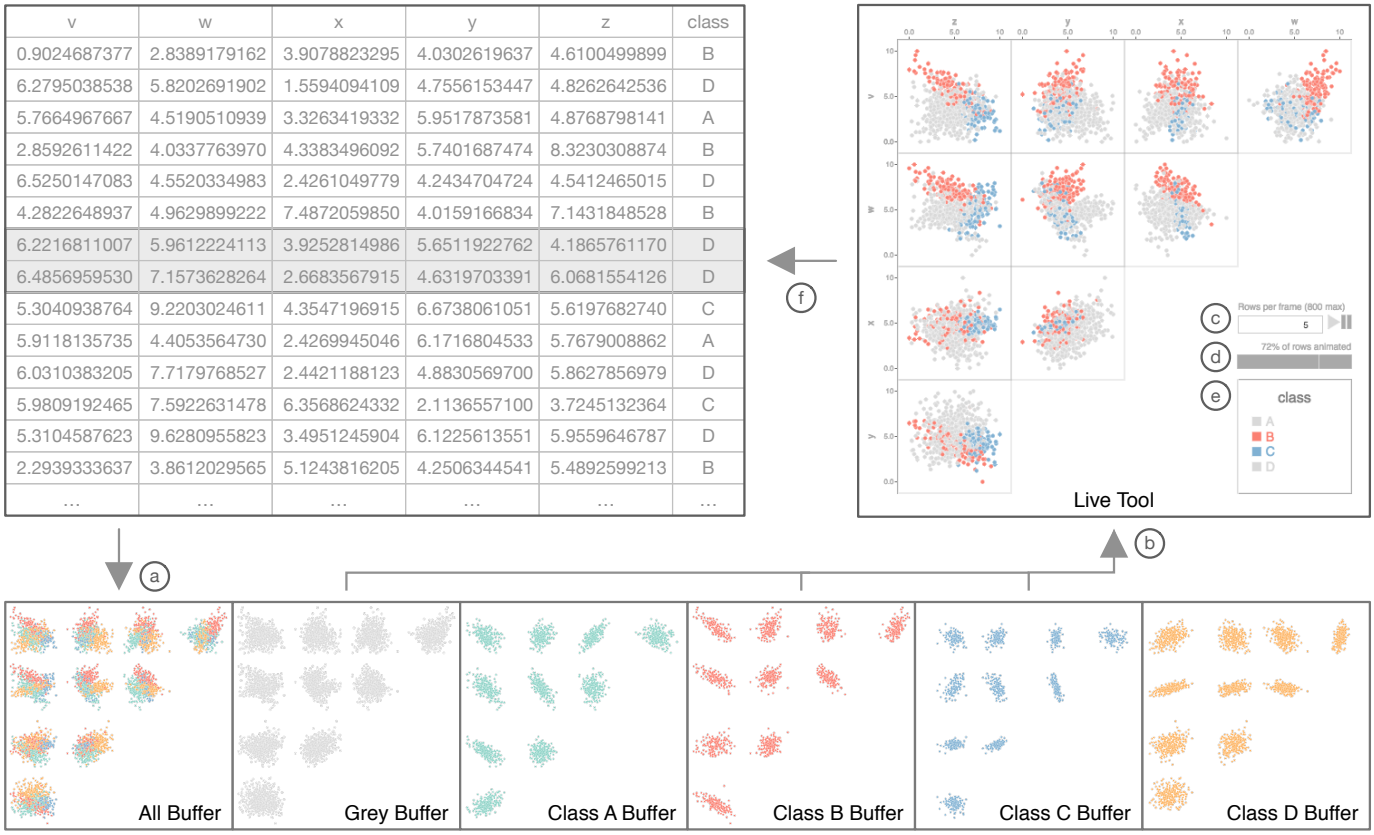

Fig. 2. Illustration of our tool and approach. **(a)** We start by drawing the points for the current row(s) in the dataset in the all, grey, and class-specific buffers. See Section 4.6 for details. **(b)** The appropriate buffers are shown in the live tool. This example shows the buffers necessary to brush the red *B* and blue *C* classes. **(c)** Users may use the spinner widget to change the number of rows animated per frame. The animation can be stopped and restarted using the play/pause buttons. **(d)** The progress bar shows the percent of rows in the dataset that have been animated, or after the first loop, the percent of rows that have been re-animated. **(e)** The legend shows the color assigned to each class. Users may click on the colored boxes to brush different classes. **(f)** Every frame, we advance to the next set of rows in the dataset and then start at (a) again to redraw those points.

Fig. 3. Snapshot of our tool animating the diamonds dataset [56]. Only the first two rows are shown, with different regions corresponding to our usage scenario in Section 5.1 highlighted. Please note some of these observations are only possible by interacting with our animated tool. **(a)** Example of an outlier. **(b)** Example of a low density and low diversity region. **(c)** Example of a low density and high diversity region. **(d)** Example of a high density and high diversity region. Also highlighted in discussion focused on the teal *Fair* class. **(e)** Region highlighted in discussion focused on the teal *Fair* class and purple *Ideal* class.

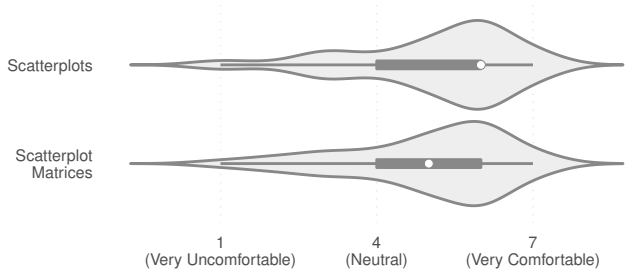

Fig. 4. In the user study, participants were asked to rate their comfort level with scatterplots and scatterplot matrices on a scale of 1 to 7, as shown above. Most participants were somewhat comfortable interpreting these plots. ***Interpretation:*** *The violin plots show the density of ratings, the box plots show the first and third quartiles, the whiskers show 1.5 times the inter-quartile range, and the white circle shows the median. This encoding is consistent across all violin plots in this paper.*

confirmed that only 1% of the points in this region belong to the *Fair* class.

- If we brush only the purple *Ideal* class, it has narrow *depth* and *table* ranges. The purple color remains persistent, indicating it is also a large proportion of points in those ranges. However, it is not until focusing on the *carat* versus *price* cell that the proportion this class represents becomes more visible (see Figure 3e). Indeed, 40% of all rows in this dataset belong to the *Ideal* class.

Next, we make some observations regarding the rendering performance of our tool with this large dataset.

5.2 Rendering Performance

We tested our tool on an iPad Air 2 tablet and 2013 iMac Desktop, both with 16 GB of memory. At 30 frames per second, it takes 3 minutes to loop through the entire diamonds dataset at 10 rows per frame and 18 seconds at 100 rows per frame. Both our tablet and desktop system were able to handle 10 rows per frame without any issue.

However, animating 100 rows at a time requires drawing 2,100 points per frame (100 points for each of the 21 cells in the SPLOM). This caused a sharp drop in framerate on both our tablet and desktop system, suggesting we may need to move to a renderer with better performance for larger datasets. As the increased rate of flickering may be overwhelming to some users, we let users control this rate based on their preferences and system capabilities. We did not observe these issues with datasets with under 10,000 rows.

We also observed performance issues with pre-rendering a dataset of this size. To pre-render all 53,940 rows in the 21 cells in our SPLOM, the our tool must pre-render 1,132,740 points total. Making matters worse, it must do this for each of the buffers used for brushing. This caused the browser to crash prior to rendering the first frame on both systems. However, there are no issues if pre-rendering is disabled. Since the first frame is always the same, we may be able to eliminate that render time by storing the first frame for each buffer as an image instead.

Overall, this scenario demonstrates that our interactive web-based tool is capable of handling datasets of this size, but with some performance tradeoffs. It also demonstrate that our approach can be applied to multi-class SPLOMs. Specifically, our tool can be used to identify outliers, regions of varying density and diversity, as well as the ranges and prevalence of different classes in the underlying dataset. Next, we evaluate not only whether our animated approach *can* alleviate overdraw, but also whether it is *better* than non-animated settings.

6 User Study

We conducted a user study to evaluate the effect of animation on multi-class scatterplot matrices as compared to static SPLOMs. The study was run as an in-lab study administered via a web application using the Experimentr [57] framework. The user study (IRB ID: 796) was reviewed by the Institutional Review Board for the Protection of Human Subjects (IRBPHS) at the University of San Francisco and approved as Exempt under 45 CFR 46.101(b).

6.1 Experimental Design

We chose to focus primarily on the impact of animation in our user study. The experiment followed a within subjects design with two conditions: static and animated SPLOMs. Specifically, we focused on being able to understand how animation impacts our ability to find regions with high density and/or diversity, the amount of time spent on finding these regions, and whether animation is too distracting for most users.

As such, we removed brushing and the ability to change the rate of animation from the tool. We also chose to use the default point encoding with opaque colored circles to isolate the impact animation made on user performance and preference, and we removed the ability for users to change this encoding. We discuss these and other design decisions in more depth in the following sections.

6.1.1 Analysis Tasks and Data Sets

There has been preliminary work identifying user tasks related to scatterplots [46]. Since we are focused on overdraw in multi-class SPLOMs, we chose tasks related to density (number of points) and diversity (number of classes). Specifically, the two tasks were:

- *Targeted Density*: Participants were asked to identify the region in a specific cell within the SPLOM with the most circles.
- *Diverse Density*: Participants were asked to identify the region in any cell within the SPLOM with the most circles and at least four different classes (colors).

The *targeted density* task focused on density without consideration of diversity, and asked participants to focus on a specific cell within the scatterplot matrix. Participants could not continue to the next task until they made a selection within the correct cell in the SPLOM. The *diverse density* task focused on both high density and high diversity. Participants could select regions in any cell within the scatterplot matrix, but had to select a region with at least four or more classes to continue. Therefore our two hypotheses for the user study are as follows:

- *Hypothesis 1*: Participants will be able to identify *targeted density* more accurately using an animated SPLOM than a static SPLOM.
- *Hypothesis 2*: Participants will be able to identify *diverse density* more accurately using an animated SPLOM than a static SPLOM.

We generated synthetic datasets tailored for each task using the pandas [39, 61], numpy [51, 59], and scikit-learn [42, 62] libraries in Python. Specifically, we generated two training datasets and one larger core dataset per task. Each dataset had 300 rows, 5 columns with values between 0 and 10, and 1 class column with 5 possible classes. We generated 6 additional datasets (3 per technique) from each of the core datasets by shuffling the rows, randomly assigning variable labels v, w, x, y, and z to columns, randomly flipping all values in the dataset, and randomly assigning class labels A, B, C, D, and E to the class numbers. The goal was to generate datasets with comparable patterns, density, and diversity, but different enough to avoid learning effects as the study progressed.

For each participant, we randomized the technique order and which datasets were used for each technique. For example, some users saw dataset 1 first using a static SPLOM, and some users saw dataset 1 last using an animated SPLOM.

6.1.2 Study Procedure

Before beginning the trials, participants completed a questionnaire about basic demographic information (such as age bracket and education level). The application then walked participants through a brief introduction to scatterplots and scatterplot matrices and asked them how comfortable they were with these techniques.

We then provided instructions on how to perform each task. Participants indicated their selected regions by dragging a black rectangle on the plot. After the instructions, participants practiced each task on both

static and animated SPLOMs before beginning the main task trials. The practice questions provided immediate feedback on how many circles and colors were currently selected. Participants then completed each task three times (three trials for static and three trials for animated). After participants completed both tasks, we then asked whether they found each technique easy to interpret, which technique they preferred, and whether they found the animation distracting.

6.2 Participants

We recruited 69 university students to participate in the study. To be eligible to participate in the study, participants had to confirm they were 18 or older, did not have a form of color blindness that interfered with their ability to interpret the class colors, and did not have photosensitive epilepsy.

Approximately one third of participants identified as female and the other two thirds identified as male. Over 78% of participants were under 24 years old, and the remaining 22% of participants were 25 to 44 years old. Participants took between 6 and 26 minutes to complete the study (about 15 minutes on average).

Participants provided ratings about their comfort level with both scatterplots and scatterplot matrices. Figure 4 shows a summary of the participants comfort levels. Most participants were moderately comfortable interpreting the plots.

6.3 Results

We analyzed the results in terms of task performance as well as for subjective preference.

6.3.1 Task Performance

Both the *targeted density* and *diverse density* tasks asked participants to select a region with the most circles (with the first specifying a cell, and the second specifying a number of classes). To assess task performance, we considered the average number of points in the selected regions and the amount of time taken to complete the tasks per participant. See Figure 5 for an overview of those results.

We conducted statistical tests for the performance results for each task to test the hypotheses that the animated SPLOMs would enable better performance than traditional SPLOMs. For the *targeted density* task, neither task time nor the number of identified points were normally distributed, so the measures did not meet the assumptions for parametric testing. We therefore opted for nonparametric Friedman tests for statistical comparison of the animated and static SPLOMs.

To test hypothesis 1, we examined the number of points identified, and found a significant effect with $\chi^2(1) = 10.88$ and $p < 0.001$. Participants identified regions with significantly more points when using the animated SPLOMs ($M = 60.63$, $SD = 13.24$) compared to the static SPLOMs ($M = 56.22$, $SD = 12.24$), supporting hypothesis 1.

Regarding task completion time for the *targeted density* task, each trial took approximately 23 seconds to complete. No evidence of a difference was detected ($\chi^2(1) = 0.71$, $p = 0.40$) between animated and static SPLOMs.

To test hypothesis 2, we examined the number of points identified from the *diverse density* task, assumptions of sphericity and normality were met for parametric testing, so we used a repeated measures ANOVA (analysis of variance) test. The effect was significant with $F(1,68) = 18.97$ and $p < 0.001$, showing more points were identified with animated ($M = 42.65$, $SD = 6.53$) than with static SPLOMs ($M = 39.45$, $SD = 5.98$), supporting hypothesis 2.

Average task completion time for the *diverse density* task was approximately 21 seconds per trial. Completion times for this task were not normally distributed, so we again used a Friedman test. No evidence of a difference was found, with results yielding $\chi^2(1) = 2.45$ and $p = 0.12$ between animated and static SPLOMs.

These results provide evidence in support of the hypotheses that animated SPLOMs are better than static SPLOMs for discriminating point density. The results also failed to detect significant differences in the amount of time needed to interpret the different types of SPLOMs.

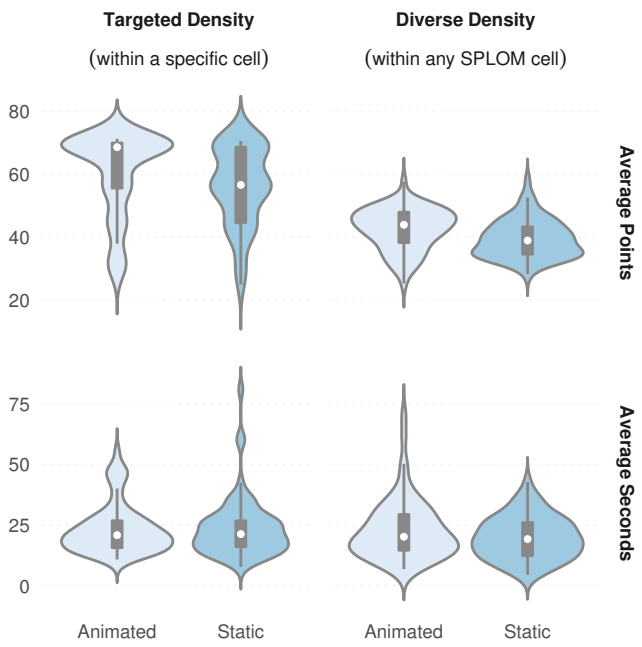

Fig. 5. Performance results from our user study. See Figure 4 for how to interpret these plots. **Top**: The average number of points selected by participants for each task and technique. As discussed in Section 6.3.1, participants were able to select denser regions using animated versus static SPLOMs on average. The effects were statistically significant for both tasks. **Bottom**: The average seconds it took to complete the tasks per participant. No evidence of difference was detected between techniques for either task.

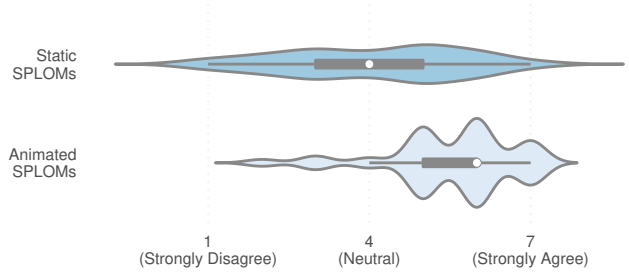

Fig. 6. Users were asked to rank how strongly they agreed with the statement that each technique (static versus animated SPLOMs) was easy to interpret. The animated plots had statistically significantly higher ratings than static. See Figure 4 for how to interpret these plots.

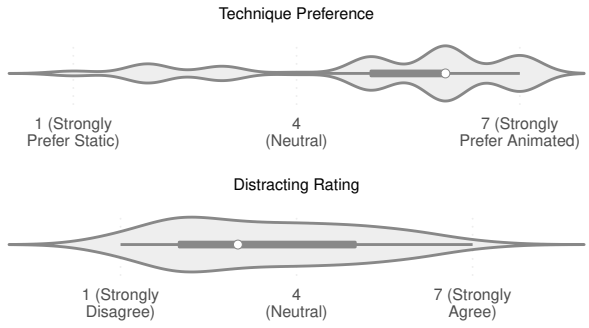

Fig. 7. Users were asked to rate their preference between static and animated SPLOMs, and whether they agreed with the statement that animation is distracting. Most users preferred animated but some found it somewhat distracting. See Figure 4 for how to interpret these plots.

6.3.2 Preferences

For subjective measures, participants found animated SPLOMs easier to interpret on average than static (see Figure 6). Due to the ordinal nature of interpretability ratings, we compared ratings for static and animated SPLOMS with a Friedman test. The animated plots had significantly higher ratings with $\chi^2(1) = 16.29$ and $p < 0.001$. These responses align with the task performance results, suggesting that participants were aware that they could more easily interpret region density with the animated SPLOMS.

Participant responses also clearly indicated a strong overall preference for the animated SPLOMS over the static variants (see Figure 7, top). Of the 69 participants, 77% percent preferred animation. Despite the clear preference and superior interpretability of animated SPLOMS, some participants found the animation distracting (see Figure 7, bottom). However, only 30% of participants indicated agreement (at any level above neutral) that animation was distracting during the study.

7 Discussion

Our interactive proof-of-concept tool and usage scenario demonstrates this approach is capable of handling datasets with over 50,000 rows (or over a million points total) in the browser, but there are performance tradeoffs as described in Section 5.2. Many of these tradeoffs can be alleviated by optimizing our tool or moving to settings with better rendering performance.

Our user study results in Section 6.3 provide strong evidence that our approach is successful at alleviating overdraw in multi-class SPLOMs. Participants in our user study performed better using animated versus static SPLOMs, confirming that animation helps distinguish region density in SPLOMs with high overdraw without the animation being too distracting. The study also did not produce evidence that animation increased the amount of time it took participants to complete those tasks. Furthermore, participants found animated SPLOMs easier to interpret and preferred them over static SPLOMs. That our novel approach using animation did better in both performance and interpretability in a comparable amount of time as a static SPLOM is a positive and promising result.

However, the benefits were not universal for all participants in our study. Approximately 29% of participants on the *targeted density* task (selecting a dense region in a specific cell) and 28% on the *diverse density* task (selecting a dense and diverse region in any cell) performed worse on average with the animated plot, and a similar number of participants (30%) found animation at least somewhat distracting. We plan to further explore the relationship between how distracting a viewer finds animation, the rate of animation and overdraw, the amount of time spent training, and overall performance in the future. For now, these results highlight the need to allow viewers to disable or adjust the rate of animation (which our tool fully supports).

As strong as these results are, we also cannot yet conclude that animation is the best approach for alleviating overdraw—only that it performs better than static for our specific encoding, datasets, and tasks. As we discussed in Section 3, there has been little attention in existing literature on overdraw in multi-class SPLOMs. Now that we know animation is a viable approach to alleviating overdraw, we can do a larger comparative evaluation study to explore how best to apply existing approaches to this setting and to determine under which circumstances one technique outperforms others.

In fact, these results open up many other interesting directions for further study. For example, we can explore how to optimize the rate of animation based on user perception, similar to the work already done on optimizing opacity [12]. We also chose to focus on multi-class SPLOMs because there is little research focused on this particular scenario, but animation may also be useful for single-class scatterplots. It would also be interesting to explore if animation always has a positive impact when combined with other encodings (such as alpha and color blending), and to explore how combining animation with existing sampling approaches affects performance outcomes.

8 Conclusion

This paper introduces a simple animated approach to alleviate overdraw in multi-class scatterplot matrices. The strengths of this approach are:

- This approach works with multi-class scatterplot matrices, which are one of the hardest cases for dealing with overdraw efficiently.
- Our proof-of-concept web-based interactive tool is capable of handling datasets that require rendering of over a million points, although there are some performance tradeoffs for large datasets.
- Participants perform better at density and diversity related tasks using animated SPLOMs over static SPLOMs, with no significant difference in how long it takes to complete those tasks.
- Participants find animated SPLOMs more interpretable and prefer them over static SPLOMs, although some find animation somewhat distracting.

However, this is only the beginning of a much longer line of inquiry. Many of the existing techniques for alleviating overdraw focus on either single-class settings or on scatterplots instead of SPLOMs. Much more study is needed on how best to alleviate overdraw in multi-class SPLOMs, and on the role animation plays in those solutions.

Acknowledgments

This work was funded by the Faculty Development Fund from the College of Arts and Sciences at the University of San Francisco.

References

[1] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1428–1435, 2008. doi: 10.1109/tvcg.2008.119

[2] L. Bartram, C. Ware, and T. Calvert. Moticons: Detection, distraction and task. *International Journal of Human-Computer Studies*, 58(5):515–545, May 2003. doi: 10.1016/S1071-5819(03)00021-1

[3] P. Baudisch, D. Tan, M. Collomb, D. Robbins, K. Hinckley, M. Agrawala, S. Zhao, and G. Ramos. Phosphor: Explaining transitions in the user interface using afterglow effects. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, pp. 169–178. ACM, New York, NY, USA, 2006. doi: 10.1145/1166253.1166280

[4] B. B. Bederson and A. Boltman. Does animation help users build mental maps of spatial information? In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, INFOVIS '99, pp. 28–35. IEEE Computer Society, Washington, DC, USA, 1999.

[5] E. Bertini and G. Santucci. Quality metrics for 2d scatterplot graphics: Automatically reducing visual clutter. In A. Butz, A. Krüger, and P. Olivier, eds., *Smart Graphics*, vol. 3031 of *Lecture Notes in Computer Science*, pp. 77–89. Springer Berlin Heidelberg, 2004. doi: 10.1007/978-3-540-24678-7_8

[6] E. Bertini and G. Santucci. Give chance a chance: Modeling density to enhance scatter plot quality through random data sampling. *Information Visualization*, 5(2):95–110, June 2006. doi: 10.1057/palgrave.ivs.9500122

[7] I. Borg and P. J. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Series in Statistics. Springer-Verlag New York, 2005. doi: 10.1007/0-387-28981-X

[8] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec 2011. https://d3js.org/. doi: 10.1109/TVCG.2011.185

[9] D. B. Carr, R. J. Littlefield, W. Nicholson, and J. Littlefield. Scatterplot matrix techniques for large *N*. *Journal of the American Statistical Association*, 82(398):424–436, June 1987. doi: 10.2307/2289444

[10] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Chapman and Hall/Cole Publishing Company, 1983.

[11] H. Chen, W. Chen, H. Mei, Z. Liu, K. Zhou, W. Chen, W. Gu, and K.-L. Ma. Visual abstraction and exploration of multi-class scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1683–1692, 2014. doi: 10.1109/tvcg.2014.2346594

[12] J. Choo, C. Lee, C. K. Reddy, and H. Park. UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1992–2001, December 2013. doi: 10.1109/TVCG.2013.212

[13] W. C. Cleveland and M. E. McGill. *Dynamic Graphics for Statistics*. CRC Press, Inc., Boca Raton, FL, USA, 1st ed., 1988.

[14] W. S. Cleveland and R. McGill. The many faces of a scatterplot. *Journal of the American Statistical Association*, 79(388):807–822, 1984.

[15] P. Craig, J. Kennedy, and A. Cumming. Animated interval scatter-plot views for the exploratory analysis of large-scale microarray time-course data. *Information Visualization*, 4(3):149–163, 2005. doi: 10.1057/palgrave.ivs.9500101

[16] T. N. Dang, L. Wilkinson, and A. Anand. Stacking graphic elements to avoid over-plotting. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1044–1052, 2010. doi: 10.1109/tvcg.2010.197

[17] D. DiBiase, A. M. MacEachren, J. B. Krygier, and C. Reeves. Animation and the role of map design in scientific visualization. *Cartography and Geographic Information Systems*, 19(4):201–214, 1992. doi: 10.1559/152304092783721295

[18] M. E. Doherty, R. B. Anderson, A. M. Angott, and D. S. Klopfer. The perception of scatterplots. *Perception & Psychophysics*, 69(7):1261–1272, 2007. doi: 10.3758/bf03193961

[19] M. Eisemann, G. Albuquerque, and M. Magnor. A nested hierarchy of localized scatterplots. In *2014 27th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 80–86. IEEE Computer Society, Los Alamitos, CA, 2014. doi: 10.1109/sibgrapi.2014.14

[20] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 2007. doi: 10.1109/tvcg.2007.70535

[21] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1148, 2008. doi: 10.1109/tvcg.2008.153

[22] D. Feng, L. Kwock, Y. Lee, and R. Taylor. Matching visual saliency to confidence in plots of uncertain data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6), 2010. doi: 10.1109/tvcg.2010.176

[23] M. Friendly and D. Denis. The early origins and development of the scatterplot. *Journal of the History of the Behavioral Sciences*, 41(2):103–130, 2005. doi: 10.1002/jhbs.20078

[24] M. Gleicher, M. Correll, C. Nothelfer, and S. Franconeri. Perception of average value in multiclass scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2316–2325, 2013. doi: 10.1109/tvcg.2013.183

[25] A. L. Griffin, A. M. MacEachren, F. Hardisty, E. Steiner, and B. Li. A comparison of animated maps with static small-multiple maps for visually identifying space-time clusters. *Annals of the Association of American Geographers*, 96(4):740–753, December 2006. doi: 10.1111/j.1467-8306.2006.00514.x

[26] M. C. Hao, U. Dayal, R. K. Sharma, D. A. Keim, and H. Janetzko. Variable binned scatter plots. *Information Visualization*, 9(3):194–203, 2010. doi: 10.1057/ivs.2010.4

[27] M. A. Harrower and C. A. Brewer. ColorBrewer.org: An online tool for selecting color schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[28] C. Healey and J. Enns. Attention and visual memory in visualization and computer graphics. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1170–1188, July 2012. doi: 10.1109/TVCG.2011.127

[29] C. G. Healey, K. S. Booth, and J. T. Enns. High-speed visual estimation using preattentive processing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3(2):107–135, June 1996. doi: 10.1145/230562.230563

[30] C. G. Healey and D. E. Huber. Visualizing data with motion. 00:67, 2005. doi: 10.1109/VIS.2005.125

[31] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, November 2007. doi: 10.1109/TVCG.2007.70539

[32] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag New York, 2002. doi: 10.1007/b98835

[33] D. A. Keim, M. C. Hao, U. Dayal, H. Janetzko, and P. Bak. Generalized scatter plots. *Information Visualization*, 9(4):301–311, 2010. doi: 10.1057/ivs.2009.34

[34] M. Krzywinski and B. Wong. Points of view: Plotting symbols. *Nature Methods*, 10(6):451–451, May 2013. doi: 10.1038/nmeth.2490

[35] D. J. Lehmann, G. Albuquerque, M. Eisemann, M. Magnor, and H. Theisel. Selecting coherent and relevant plots in large scatterplot matrices. *Computer Graphics Forum*, 31(6):1895–1908, 2012. doi: 10.1111/j.1467-8659.2012.03069.x

[36] J. Li, J.-B. Martens, and J. J. van Wijk. A model of symbol size discrimination in scatterplots. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pp. 2553–2562. ACM, New York, NY, USA, 2010. doi: 10.1145/1753326.1753714

[37] J. Matejka, F. Anderson, and G. Fitzmaurice. Dynamic opacity optimization for scatter plots. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 2707–2710. ACM, New York, NY, USA, 2015. doi: 10.1145/2702123.2702585

[38] A. Mayorga and M. Gleicher. Splatterplots: Overcoming overdraw in scatter plots. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1526–1538, 2013. doi: 10.1109/tvcg.2013.65

[39] W. McKinney. Data structures for statistical computing in Python. In S. van der Walt and J. Millman, eds., *Proceedings of the 9th Python in Science Conference*, pp. 51–56, 2010.

[40] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. In *ACM Transactions on Graphics*, vol. 22, pp. 453–462. ACM, New York, NY, USA, July 2003. doi: 10.1145/882262.882291

[41] S. E. Palmer. *Vision Science: Photons to Phenomenology*. MIT Press, 1999.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Animated visualization of multiple intersecting hierarchies. *Information Visualization*, 1(1):50–65, March 2002. doi: 10.1057/palgrave/ivs/9500002

[44] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, November 2008. doi: 10.1109/TVCG.2008.125

[45] C. Roda. *Human Attention in Digital Environments*. Cambridge University Press, New York, NY, USA, 2011.

[46] A. Sarikaya and M. Gleicher. Tasks to tease apart scatterplot design decisions. Poster, IEEE VIS 2016, 2016.

[47] M. Sedlmair, T. Munzner, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2634–2643, 2013. doi: 10.1109/tvcg.2013.153

[48] J. Shearer, M. Ogawa, K.-l. Ma, and T. Kohlenberg. Pixelplexing: Gaining display resolution through time. In *IEEE Pacific Visualization Symposium (PacificVIS '08)*, pp. 159–166. IEEE, 2008. doi: 10.1109/pacificvis.2008.4475472

[49] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: Can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, October 2002. doi: 10.1006/ijhc.2002.1017

[50] L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

[51] S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, March 2011. doi: 10.1109/MCSE.2011.37

[52] C. Ware and R. Bobrow. Supporting visual queries on medium sized node-link diagrams. *Journal of Information Visualization*, 4(1):49–58, March 2005. doi: 10.1057/palgrave.ivs.9500090

[53] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009.

[54] ColorBrewer. http://colorbrewer2.org/.

[55] D3.js JavaScript Library. https://d3js.org/.

[56] Diamonds dataset. http://docs.ggplot2.org/current/diamonds.html.

[57] Experimentr. https://github.com/codementum/experimentr.

[58] ggplot2 R Package. http://ggplot2.org/.

[59] numpy Python Package. http://www.numpy.org/.

[60] P5.js JavaScript Library. http://p5js.org/.

[61] pandas Python Package. http://pandas.pydata.org/.

[62] scikit-learn Python Package. http://scikit-learn.org/.