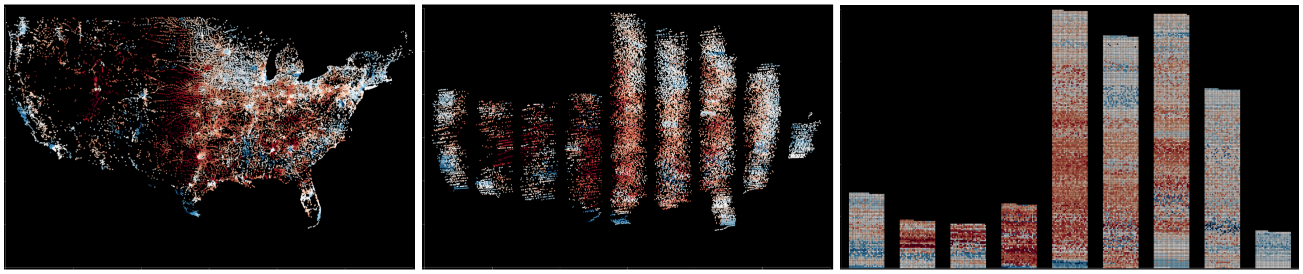
A Unifying Framework for Animated and Interactive Unit Visualizations

Steven M. Drucker and Roland Fernandez



Abstract—We use the term Unit Visualizations to describe a class of visualizations that explicitly represent every row in a data set. They have been around in one form or another for hundreds of years, usually in static form (e.g. tallies, scatterplots, Dot Plots, Unit Charts, Pixel Charts, or Isotypes.) We characterize their design space and propose a unifying framework that can produce common types of Unit Visualizations. In addition, we introduce SandDance, a tool built to explore the effectiveness of animating between and interacting with unit representations when analyzing and presenting multidimensional datasets.

Index Terms—Visualization models, Multidimensional Data, Particles, Storytelling

1 Introduction

We use the term Unit Visualizations to describe a class of visualizations where each row of the data-table has a single graphic representation which we refer to as a unit. Unit Visualizations stand in contrast to aggregate visualizations (like Bar Charts) where a single graphic element represents a statistic (e.g. count, sum, mean) of a group of rows. Unit Visualizations have been called by a range of names, including Unit Charts, to Dot Plots, Tallies, Pixel Charts or Isotypes) [13, 32, 21, 29, 22].

Unit Visualizations are often used in simple, static, info-graphic type illustrations, perhaps because of the straightforward way that a particular graphical element is tied to the semantic meaning of a row (so that an icon of a person might represent a row of data that represents a person). Some have commented that they are simplistic [13] but they have several potential advantages, especially when used in interactive systems.

1.1 Advantages

The key aspect of unit visualizations is that they maintain a one-to-one correspondence between rows in a data table with representations (or units) while still allowing visual attributes (such as color or size) to be mapped onto the individual units.

The forest and the trees: Since all rows are represented on the screen individually, they can simultaneously show overall patterns as well as individual outliers. Other common forms of representations, such as bar charts, histograms, pie charts, or even tree-maps require a level of aggregation of the data such that multiple rows are combined and that combination is then mapped onto a shape shown in the chart. Strictly speaking, some Unit Visualizations *do* perform some aggregation, but only in that marks are proportional to the number of data rows (e.g. A single mark might represent 1000 people in a census.) Aggregation can hide outlying data, or have outlying data unduly influence the aggregate representation.

• *Steven M. Drucker and Roland Fernandez are with Microsoft Research. E-mail: sdrucker@microsoft.com, rfernandez@microsoft.com.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Semantic Constancy: The association between a row and a unit is maintained even when the position or visual attributes (color, size, shape) of the unit changes. While aggregation is powerful, it requires a certain level of abstraction —the bars may represent different quantities depending on what aggregation function is being performed (sums, means, maximums, etc. on a particular attribute). This power can require a certain amount of cognitive effort to understand what a particular representation might mean. Even when people are experienced with understanding visualizations, aggregating information can hide underlying variables that might help explain a situation (e.g. Simpson's Paradox [5]).

Direct Interaction: Individual units provide an affordance for interacting directly with each row of the data (e.g. tooltips, selection). Interaction with Unit Visualizations allows a user to directly select and find information about a single constituent unit (which is not possible for aggregate visualizations). In multiple views of the same dataset, brushing and linking is straightforward; units selected in one visualization are highlighted in another. In aggregate visualizations, it is more difficult to show correspondences between two different visualizations where the aggregation might be along different parameters.

Animation: Direct correspondence between units in different visualizations of the same datatable enable us to animate smoothly between them. Transitions need, therefore, to only update between before and after positions of every element since they are all present in both visualizations. A straightforward, linear interpolation of positions can be used, or more sophisticated transitions, including bundled trajectories [10], or dimension adding/rotating such as ScatterDice [11] or temporal distortions [8]. Since individual items can be selected in one view, they can be tracked as they transition to a new view, which can also help users understand patterns in the data. Animations between aggregate visualizations require shapes to distort, break apart, or come together or even do a simple cut or cross fade when there are no clear, direct relationships between the visualizations. These can make it harder to undertand sequences [15].

These advantages are especially important given that multiple views of a data set (or sequences of visualizations) are often more effective for both exploring and conveying stories about data [26, 17] than a single, static view.

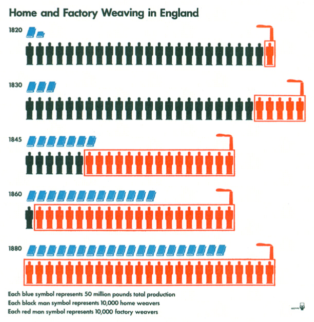


Fig. 1. ISOTYPES, from Neurath, an early form of Unit Visualization used in Infographics[22]

1.2 Disadvantages

While there are many advantages, there are some potential disadvantages, both in terms of optimal interaction and implementation.

It is often easier to judge the values of aggregate bars vs. unit bars (via the y axis) and aggregate charts can be visually simple (with less clutter).

Aggregate charts provide a natural affordance for getting information about the aggregate (count, sum) while unit charts must provide an additional affordance for accessing aggregate infomration since a unit has both individual information as well as information about the particular group it is contained within.

More significantly, aggregate visualizations have the potential to scale much more readily to large data sets. They can be much quicker to display, and have no upper limit to the number of individual items that they represent, while unit visualizations reach a limit on pixel density or drawing speed of display hardware. From a representational point of view, there are only a finite amount of pixels on the screen and eventually, showing all the data might be meaningless unless data is represented by subpixel marks, in which case outliers might be missed. Opacity can be adjusted to create semi-transparent shapes which allow for more overlap, but eventually, a limit on the number of shapes that can be clearly discerned will be reached. In addition, as the number of marks increases, transitions start to become potentially incomprehensible without more sophisticated bundling techniques [10].

Finally, in a client-server configuration, a server might calculate aggregates, requiring very little information to be passed back to a client, while unit visualizations require significantly more (per unit) information.

Because of these disadvantages, unit visualizations may not be appropriate in every circumstance, and in fact, Elmqvist & Fekete [12] discuss an overview of situations where aggregation can be useful especially for reducing clutter and scaling. Still, many disadvantages can be overcome by careful design choices and in the interaction model and the relative advantages can be compelling, especially when working with multidimensional datasets of up to 1 million rows where screen densities and client server communication latencies are less of a problem.

1.3 Contributions

After having discussed some basic advantages and disadvantages for unit visualizations, we will use the rest of this paper as follows:

We first survey the literature on the the historical use of Unit Visualizations.

We then show that a simple yet powerful framework for laying out units can generate unit versions of many common charts, including Scatter Plots, Bar and Column Charts, Box and Violin Plots, and 2D histograms (also referred to as Fluctuation Diagrams [16]). This framework also allows us to explore unit representations that are potentially useful but less commonly seen. Most importantly, this framework allows a simple, unified method of animating between and interacting with all of these representations. We subsequently introduce SandDance, a prototype that was built to explore the effectiveness of this representation for analyzing and presenting moderately sized multidimensional datasets (1K-300K rows by 4-30 dimensions) and show its utility, especially for telling data stories to novices, and report on anecdotal results from a deployment for over 50 people.

2 Related Work

The notion of having a direct correspondence between rows of data with shapes or 'marks' in a visualization has a long history. Tallies (figure 1a), are perhaps one of the simplest visualizations where a specific mark is made to correspond with a single observation and can be used as a shorthand to help count the number of observations present [24].

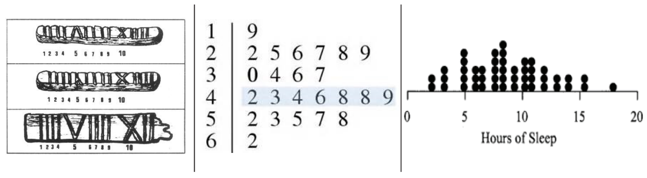


Fig. 2. Tallies, Tukey's Stem and Leaf Plots, Wilkinson's Dot Plots

Neurath [22] and others developed ISOTYPES (Fig. 1) (International System of Typographic Picture Education) in the 30s not only for signage but also for use in constructing visualizations. Tukey reformulated Tallies as stem-and-leaf plots [29], while Dot Plots [32], (figures 1b and 1c) are used to represent observations laid out along a single axis where we can immediately see the relative values of a particular variable associated with the observations.

While dot plots show a single variable associated with an observation, scatterplots show two variables associated with a single row, mapped onto the Cartesian plane. Scatterplots, one of the original unit charts, have been considered to be one of the most versatile, and useful invention in the history of statistical graphics [14] and it was estimated by Tufte in 1983 [28] that 70-80% of graphs used in scientific publications are scatterplots. Scatterplots suffer from overplotting and the framework described here finds general ways of avoiding overplotting.

Further differentiation of the values in any Unit Visualization can be accomplished by associating color, shape, and size with each of the marks.

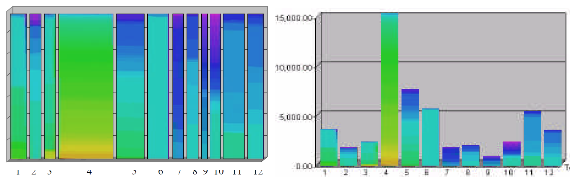


Fig. 3. Keim's Pixel Charts [21]

There has been a long history of trying to combine the best of both worlds by representing both the individual points as well as a sense of their aggregation. Perhaps the closest related to this work are Keim's Pixel Bar Charts [21] where individual items are represented as pixels grouped together within either space filling or bar chart form (figure 3). Keim describes a formalism for pixel bar charts which involves coloring, ordering and partitioning of the bars. We generalize this work, allowing units to take on a greater variety of shapes, and support hierarchically nested containers to achieve a greater number of final chart types. We also support interactive and animated transformation of charts from one representation to another as well as the interactions that Keim proposes.

Since we support nesting of containers, each with potentially different layout criteria, our work also has similar properties to Wickham's Product Plots, where combinations of rectangles with area proportional to underlying counts or probabilities are supported. Since at the base level of our work, units are laid out in groups, these groups can be often be thought of as 'mass preserving' where $Density = (Mass * Count)/Area$. Many chart layouts, as discussed below, use a notion of constant density, and as such this equation reduces to $Area = Mass * Count$ or *area* is proportional to the *count* of elements in a group. As such, many of the same analytic benefits of the Product Plot system are realized [31]. While Product Plots focus strictly on

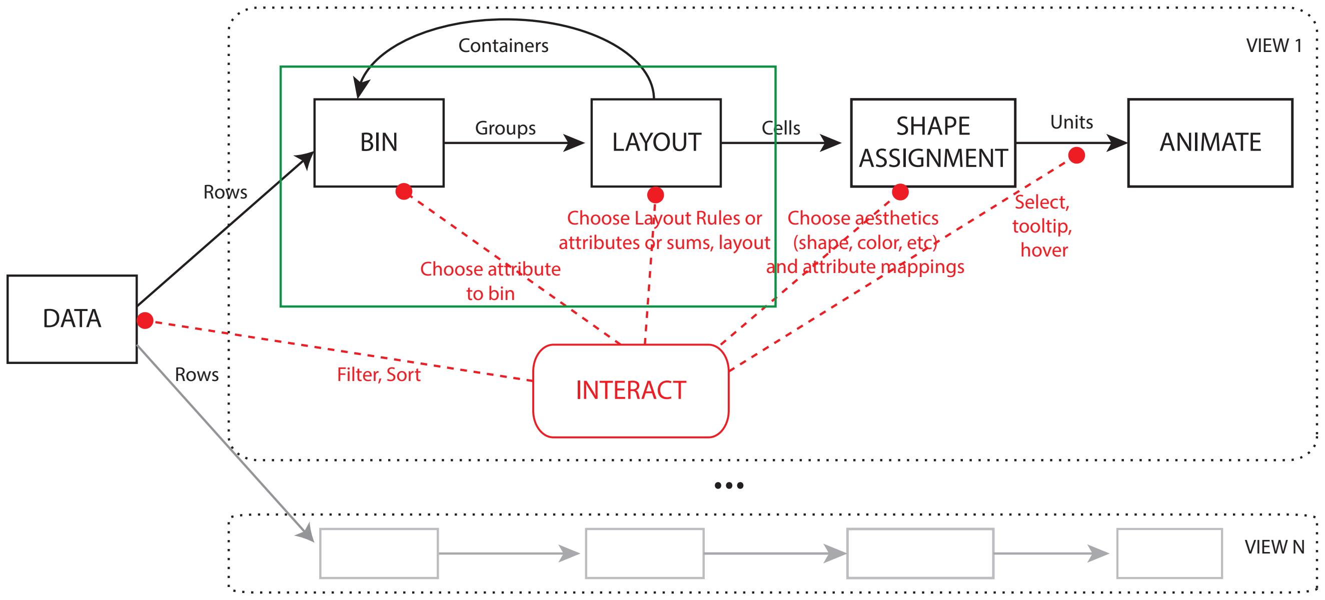


Fig. 4. The overall Unit Visualization Framework: multiple views, animation and interaction are all supported. All interaction is in red. We focus teh explanation on the binning and layout portion of the framework in green.

aggregations, Unit Visualizations show aggregations in terms of how the units are grouped on the screen.

A core notion of many of the layouts in Unit Visualization is to avoid overplotting. Dang et al [7] discussed a generalization of stacking (in 2 and 3D) for a variety of chart types including dot plots and parallel coordinates. As such, this work shares similarities, though we generalize the notion to a framework for specifically dealing with unit representations.

Other inspiration for the Unit Visualization framework comes from Wilkinson's Grammar of Graphics [33] (and Wickham's refinement of it in [30]). We use a similar notion of marks and aesthetics such as fill color, stroke, opacity, shape, and size that are assigned to the marks, but only at the unit level. Thus many levels of the grammar remain the same, but operations which change the number of rows in the data are not permitted. We also explore animated transitions and interaction which are currently not part of these grammars.

Several vertical domains have shown the usefulness of Unit Visualizations including image editing and machine learning. Histomages [6] show the benefits of using Unit Visualizations for interacting with both images and color histograms allowing convenient selection in either space. The Model Tracker visualization [3] enables outliers to be immediately identified in context by displaying all labeled and predicted results in a Unit Visualization.

Huron et al have explored the use of physical tokens to help manually construct tangible unit representations of data. They found that representing data in this way helped novices effectively build and communicate visualizations [19]. This tangible form of Unit Visualization helps illuminate how novices can construct visualizations.

Several physically based/inspired representations such as Yi et al's Dust and Magnets [35], Rzeszotarski's TouchViz [25] and Huron's Visual Sedimentation [20] all use the notion that individual rows of data that are physically represented as units on the screen, and move and are laid out using physical constraints. In our system, we do not focus on physical simulations, but instead use layout rules to achieve non-overlapping packing of containers.

Image based layout systems like MediaBrowser and PhotoMesa [9, 4]are examples of Unit Visualizations where the unit represented is an image. Most do not focus on general layout based on numerical attributes associated with the image though binning by dates or events are often used.

We stage transitions between representations (such as filtering out elements before relaying out visualizations) in a fashion similar to Heer & Robertson [15]. In general, while we primarily support linear interpolation between initial and final position of units, more sophisticated transformations such as those in ScatterDice [11] which adds another dimension orthogonal to the current dimensions, rotates to show that dimension, and then flattens to new dimensions; or bundling objects that start and end in similar locations to help minimize random motion [10] can also be used. While theoretically, any unit chart can be transitioned to any other unit chart of the same data, in our prototype system we limit the number of dimensions that can be changed at any one time via UI affordances. In this way we help achieve logical sequences that are more comprehensible [18].

Finally, many of the features of the SandDance Prototype itself explore interactions with unit charts based on systems like Spotfire & Tableau[1, 27] (simple interactive assignment of attributes to dimensions), brushing, linking, filtering on demand found in most systems.

3 Framework

The general framework for Unit Visualizations is described in this section. The overall framework is illustrated in figure 4. For purposes of this discussion, we will use the following definitions:

- *Data* is defined to represent a datatable which contains multiple *rows* with each *row* containing multiple columns (or attributes)
- *Groups* are a subset (up to and including the entire datatable) of *rows*. *Groups* can also be nested, so a group might in turn contain other groups
- *Containers* are a geometrical abstraction with a position and area in which a *group* will be laid out.
- *Cells* are a special instance of a container which are associated with one and only one *row* of the dataset
- *Units* are the graphical representation of a *row* of data. They can have visaul attributes such as color, shape, size (relative to their enclosing cell), and opacity.
- *View* is a particular visualization of a datatable; it can be linked with other views of the same data.

The goal of the framework is to create a common way of laying out, animating, and interacting with unit representations of a data-table to facilitate exploration and presentation of the data. While much like a typical information framework, we focus on a few simple rules for packing containers within other containers to eventually establish positions for all the units on the screen. In general, shape assignment, animation, and interaction is handled in the same way as many other

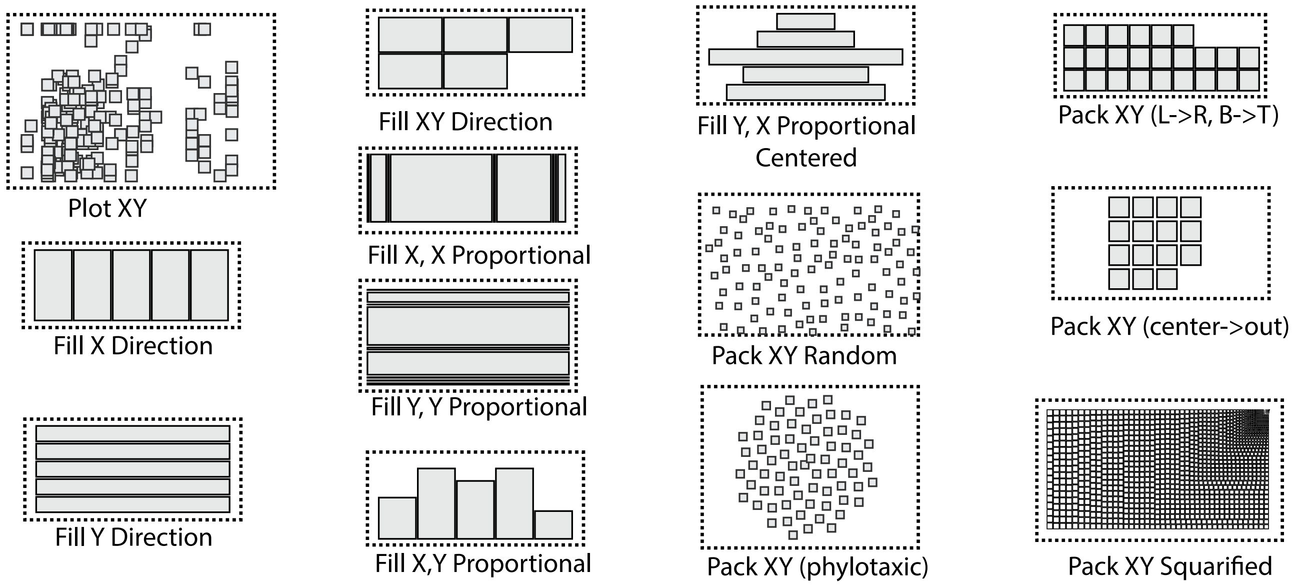


Fig. 5. Layout Rules: Some examples of different primitive packing rules can be used to fill any container. Fill expands the cell in a specified direction, while pack densley packs cells in the specified fashion. These in turn can be combined with other rules to create common unit layouts shown in 6.

existing frameworks, so in this section, we will concentrate primarily on the layout steps required to create a Unit Visualization. In the following section, we will demonstrate how this framework enables a consistent interaction and animation model.

Like UI Frameworks [23] that arrange widgets on a screen, we use a similar pattern of recursively descending a hierarchy asking children for information in order to determine how much space is necessary to display their contents. At each level of the hierarchy are groups of rows (or groups of groups). Once the lowest level of the hierarchy is reached, information about the relative statistics in each group is passed back to the root so that a uniform size for cells can potentially be enforced. The layout algorithm uses layout rules to arrange all the containers (and eventually, all the leaf containers or cells). Some layouts choose to explicitly make all cells the same size and density, for easy direct comparison of counts, while other layouts allow different sizes to support other types of queries including comparisons of sums, or proportions.

We've found that a small set of layout rules (shown in figure 5), with each rule being applied at a different hierarchy level, can generate many common and useful representations. These will be discussed in more detail below.

The following layout examples are based on the Titanic Dataset which contains one row for each of 1309 passengers. For each passenger, we have information on the gender, age, cabin class, and whether the passenger survived. Units in any visualization of this data will always represent a single passenger.

Overall layout is done with the following steps:

1. **Data is first divided into abstract groups using a grouping criterion.** For instance, we might first divide by 'cabin class' to create three groups and subsequently divide each of the groups by gender and finally divide by whether the individual survived or not, creating two groups within each of the above subgroups. Thus the dataset is divided into 12 different subgroups, each containing a number of rows from the dataset. The largest (Third Class men who didn't survive contains 417 members), while the smallest (First Class women who didn't survive contains 5). The grouping criterion can be changed by user interaction.

2. **Both rows in the database and containers are optionally sorted to determine the order of layout.** For instance, we may wish all the women to be shown in containers before men so we can sort the entire dataset by gender. Or we may wish show which cabin class has the most passengers, so we can choose to sort the containers based on the count of rows contained within each group.

3. **Groups are repeatedly laid out using one or more of the layout rules described in figure 5 above in order to create positions and sizes for containers** Layout rules fall into two broad classes. Unordered rules, for example, where layout is either random or purely based on attributes associated with the rows of data. A scatterplot is a typical example of an unordered rule. One aspect of a scatterplot is that it can frequently obscure data by over plotting. While jittering can help to some extent, other rules which purposefully avoid over plotting are useful. Ordered (or packing) rules, as a class, avoid over plotting. They partition each container into subcontainers using the following criteria: a primary direction or directions of packing; a decision of whether they are either space filling, or densely packed; alignment (horizontal or vertical) to the parent container if they're not space filling; and a decision whether the sizes are all the same or whether they are proportional - either directly to an attribute of a single row of data or based on a calculated statistic of the group of data associated with the container. Refer to figure 6 to see how different layout rules at different hierarchical levels are combined to produce various common plot types in unit form.

4. **Shapes are chosen to represent each unit (Now in the leaf nodes (cells) of all containers)**. Fill color, opacity, stroke, shape, glyph, image, labeltext, and size (relative to bounds of the unit's cell) are chosen. These can be set and changed interactively and either chosen uniformly (e.g. set all the colors to blue) or based on an attribute for each row (e.g. set color based on cabin class) or group of rows (e.g. set color based on the count of units in the parent container).

5. **All shapes are drawn on the screen.** If a previous representation is already on the screen, all the objects are animated from their old positions to their new positions. Animated attributes include color, opacity, selections, and positions.

6. **Interactions** can affect the visualization in several different ways. The data itself can be explicitly sorted or filtered to change the overall visualization. Filtering can either choose to eliminate rows from the dataset from the visualization or filtered and unfiltered data can be laid out in separate areas (the filtered items might all be moved to a separate area on the screen). Different attributes or binning criteria can be interactively specified. Thus

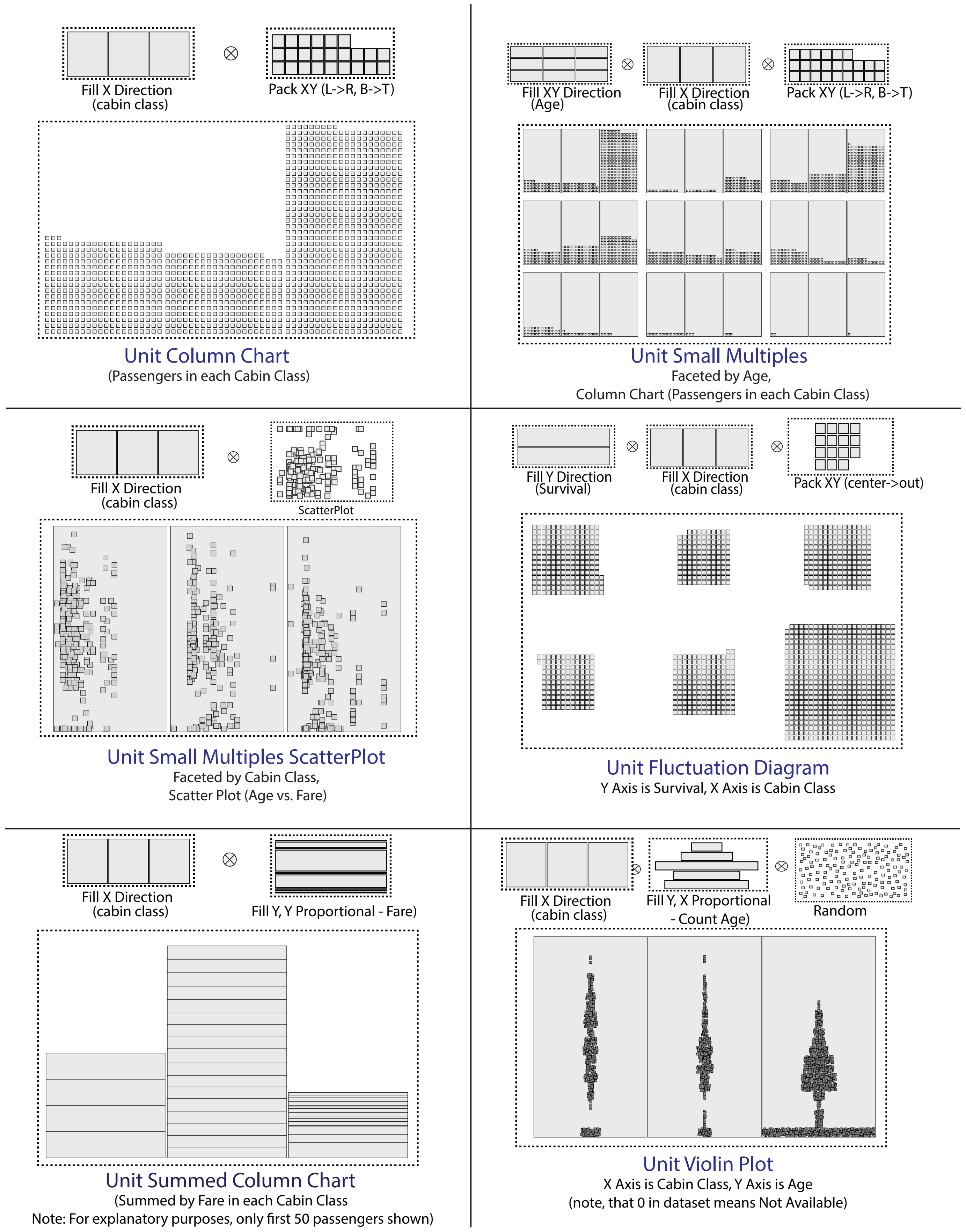
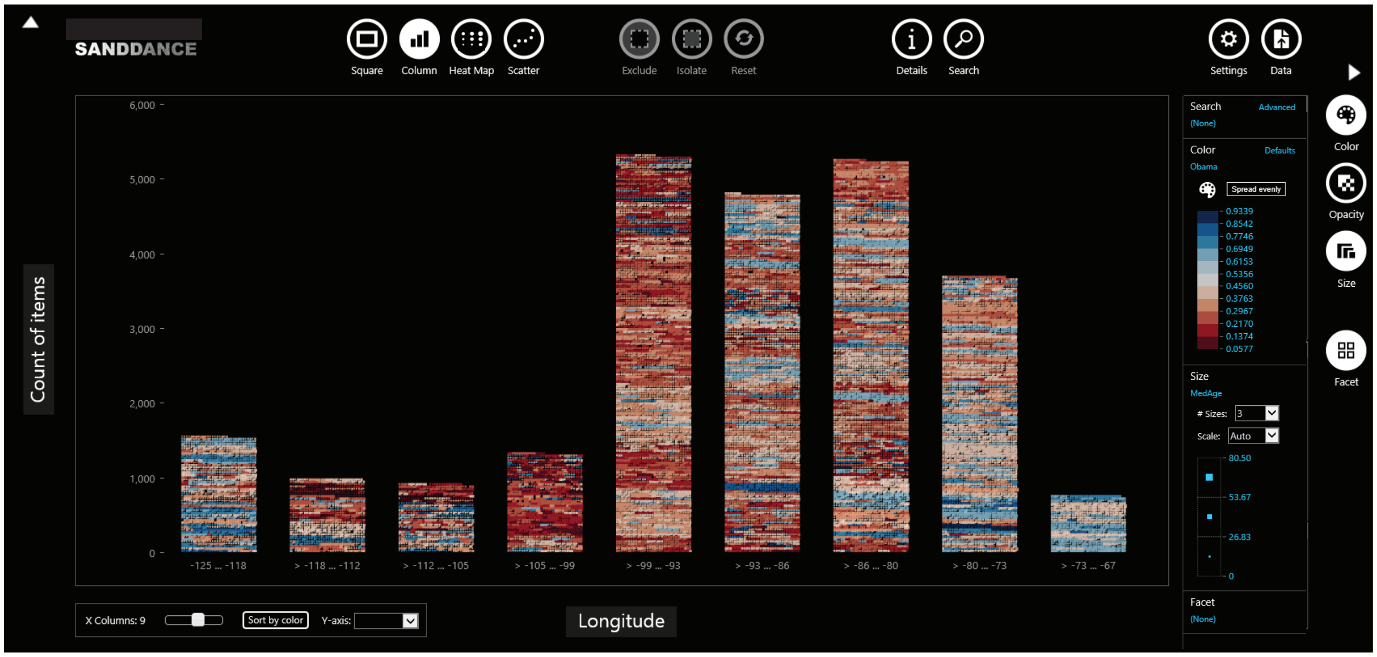


Fig. 6. Packing Primitives can be combined to produce a wide variety of Unit Visualizations including Unit Column Charts (top left), Faceted Small Multiples (top right), Faceted Scatter Plots (middle left), Fluctuation Diagrams (middle right), Summed Column Charts (bottom left) and Violin Plots (bottom right)



X Columns: 9

Y-axis:

Search: (None)

Color: Obama

Size: MedAge

Sizes: 3

Scale: Auto

Facet: (None)

Fig. 7. SandDance Prototype Unit Visualization System. Top left allows chart type selection. Top Left Middle allows filtering of the dataset, Top Right Middle allows information and searching, and Top Right are settings and load data. Right side are interactive legends for color, size, opacity, and facet. Axis labels are selectable and allow attribute selection.

we can choose to bin by Cabin Class, Gender or Age; each of which would produce different visualizations of the same data. Since age is a continuous variable rather than categorical, we can also choose how many bins to use. We can choose the layout type, and what parameters are being mapped in certain layouts (x and y axis for scatterplots, attribute to sum by in summed column charts). Shape, color, and other aesthetic attributes can be assigned to the units themselves. And finally, selection can supported in a variety of methods - either through direct selection on the units themselves or through other mechanisms that we will discuss in the prototype implementation section. Selection can alter the visual attributes (like changing the color of selected units to a unique selection color or changing unselected units' opacities).

4 Prototype Implementation: SandDance

A prototype was built to explore the benefits of using Unit Visualizations (figure 4). The prototype was implemented in HTML5 and typescript, using WebGL to enable animated transitions on datasets with up to 1 million points. While the browser can render 1 million points at 30 fps, other considerations make the current maximum dataset size considerably smaller. Memory usage maxes out in 32 bit browsers on sets of over 500K with multiple columns of attributes. Sorting and filtering within javascript, while reasonably quick, can start to take significant amounts of time ($> 500ms$) on datasets with more than 300K items.

The interface was designed so that most operations could be easily accessed via touch, though explicit text search capabilities still require keyboard entry. Conventional mouse based interaction is also fully supported. We surfaced a number of capabilities in the SandDance prototype to make it easy to use Unit Visualizations to do analysis and presentations. Instead of having people construct Unit Visualizations from combinations of primitives (though a separate tool was built to explore new combinations of the primitives), we instead focused on ease of use for exploring and presenting data. Users could pick from amongst several basic chart types (square; bar charts; column charts; 3D column charts; and Fluctuation Diagrams which are essentially 2D histograms or heat maps and scatter plots). The axes can be clicked on to specify columns, and tools at the right permit mapping columns to facets and visual attributes such as color, shape and size.

Figure 12 show a typical exploratory or explanatory sequence. A scatterplot of the 2010 census data from the continental United States is presented. This data was combined with results from the 2010 election. Units represent Census Designated Places (CDPs - approximately equal to a County). They are first plotted in a Scatter Plot (Longitude vs. Latitude) and colored by the percentage in the county voting for Obama (A). We then switch to a Unit Column Chart by Longitude (B). Relative Latitude is preserved, so the features like the Democrat leaning counties in southern Texas can be observed, but total counties per Longitude bin are hard to judge. We can sort the dataset by percentage voted for Obama within each CDP (C). Now it is relatively easy to see the number of CDPs in each longitude band that voted in a similar fashion. We can then create a summed column chart by the total population of each CDP (D). It's easy to see the CDPs with larger numbers of people tended to vote Democratic. We then bin by percent voted for Obama (E). And finally, switch to 2 bins to see an indication why Obama won the 2010 election. Note well then we are not looking at the distributions within each CDP, we're considering each CDP as a unit (essentially we look only at the percentage of votes that Obama received in each CDP). Also note, that we're not considering Electoral Votes, but using population as a proxy for them.

4.1 Selections

There is extensive support for the powerful selections that Unit Visualizations make possible. In the most direct form, units can be selected by directly clicking on a unit, or dragging a rectangle around a number of units to select them all. All the units within a group can be selected by clicking on a label that represents that aggregate (for instance, if in the titanic data, a unit column chart is showing the number of passengers in each class, the class name under each bar can be clicked on to select all passengers in that class).

Interactive Legends (figure 8) are used for 3 purposes: reflecting which columns are currently being mapped to color, size, and shape of a unit; changing what is being mapped to the color, size and shape; and selecting all units with a particular mapping: (eg clicking on the blue color automatically selects all the units that were currently being displayed in blue).

Textual search can be used to find any records that match a particular text (either anywhere in its contents, within a specified column, or only at the beginning of a specified column). Matching units aare selected. In addition, a conventional histogram along that attribute was brought up to allow quick selection by bins.

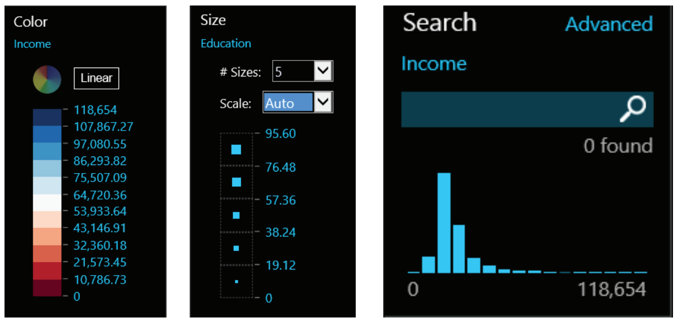
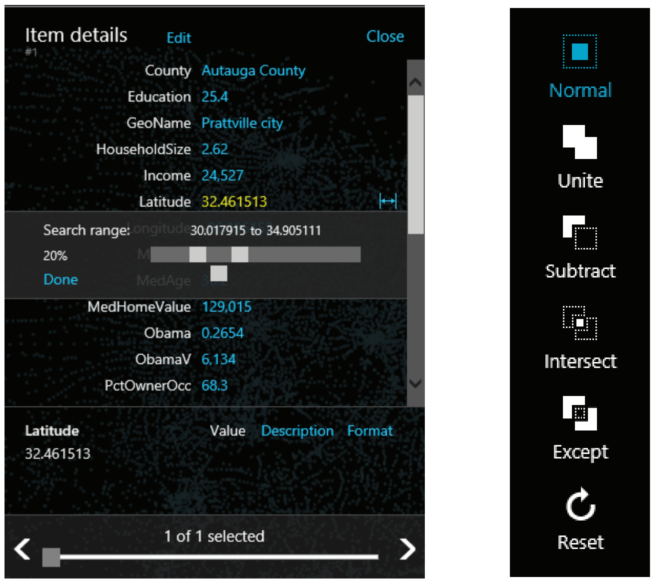


Fig. 8. Interactive legends for mapping and selecting based on attributes. Color (left), Size (middle), Search (right)


Item details Edit Close
#1

County	Autauga County
Education	25.4
GeoName	Prattville city
HouseholdSize	2.62
Income	24,527
Latitude	32.461513

Search range: 30.017915 to 34.905111
20%
Done

MedHomeValue	129,015
Obama	0.2654
ObamaV	6.134
PctOwnerOcc	68.3

Latitude Value Description Format
32.461513

1 of 1 selected

Fig. 9. Information on each item can be selected and similar items can be selected along any attribute (left). Selection behavior can be modified to achieve boolean selection combinations (right)

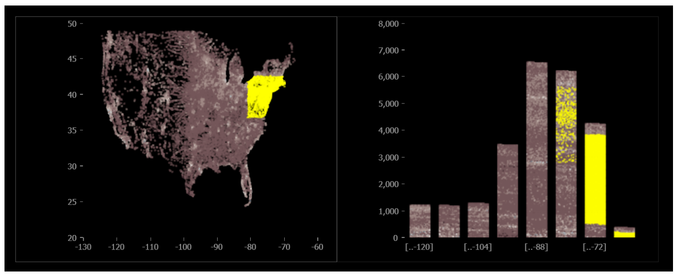


Fig. 10. Brushing and linking between multiple views (potentially in multiple browser sessions across multiple users) is supported.

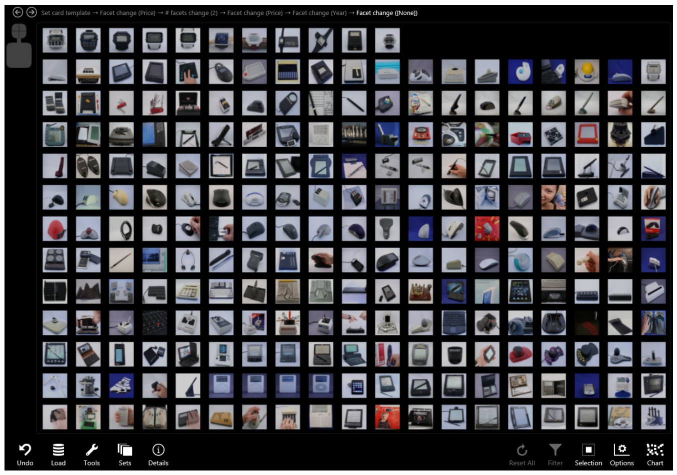


Fig. 11. Units represented as cards or images allow direct visual feedback as well as unit visualization layouts.

Item details can be brought up for the selected items (figure 9). Similar items can be selected by clicking on an attribute (categorical attributes are directly selected while numerical categories allows a range to be interactively selected).

While keyboard modifiers could be used to modify how selections were adjusted in conventional ways, we also support selection modes to facilitate touch only interaction. When in a mode, new selections are combined with old selections using the specified Boolean operator. Using the available selection methods, it's very easy to select, for instance, all the people that survived the sinking of the titanic who paid less than a certain amount for their fare.

4.1.1 Filtering of Selections

Once a selection has been made, the associated units can either be excluded or isolated (everything else filtered out of the view). We use a staged transition [15] so that items are first filtered from the view and subsequently rearranged to make maximum use of the available space. We chose to eliminate units from the screen in our first implementations, though they can also be moved to a separate area on the screen for units that have been removed from the overall layout.

4.1.2 Brushing and Linking Selections

The prototype supports multiple, linked views. Selections in one view are reflected in another view. See figure 10. Since the prototype is implemented as a web app, a lightweight, state-synchronization service has been implemented so that anybody can join a sharing session by opening up the appropriate URL. Sessions can be linked to update all changes including views, color mappings, filters, and selections; or just selections so that different views of the data can be explored in multiple windows by multiple participants.

4.2 Animation

When a new view is displayed, every item is animated between its original position and a new position and between the previous unit's attributes like color and opacity and its new ones. As mentioned above, we use staged animations when filtering out items. We also maintain selections so that users can track items and see whether they move together or to different parts of the resultant diagram. In order to help the animations be more coherent, the interface is designed to allow only one dimension of the data to be changed at a time. So, for instance, when moving from a Scatter Plot to a Column Chart, the x axis for the column chart is chosen to be the same axis as is currently being displayed in Scatter Plot. In addition, the data is sorted by the Y axis attribute of the scatterplot so that the order of items in the columns best matches the layout in the scatterplot. Figure 12 shows a typical exploratory sequence within SandDance on a census/voting dataset.

Animations can also be staggered to assist users in understanding where different groups end up going. While animations help users understand the overall structure of changes, there is no way that users

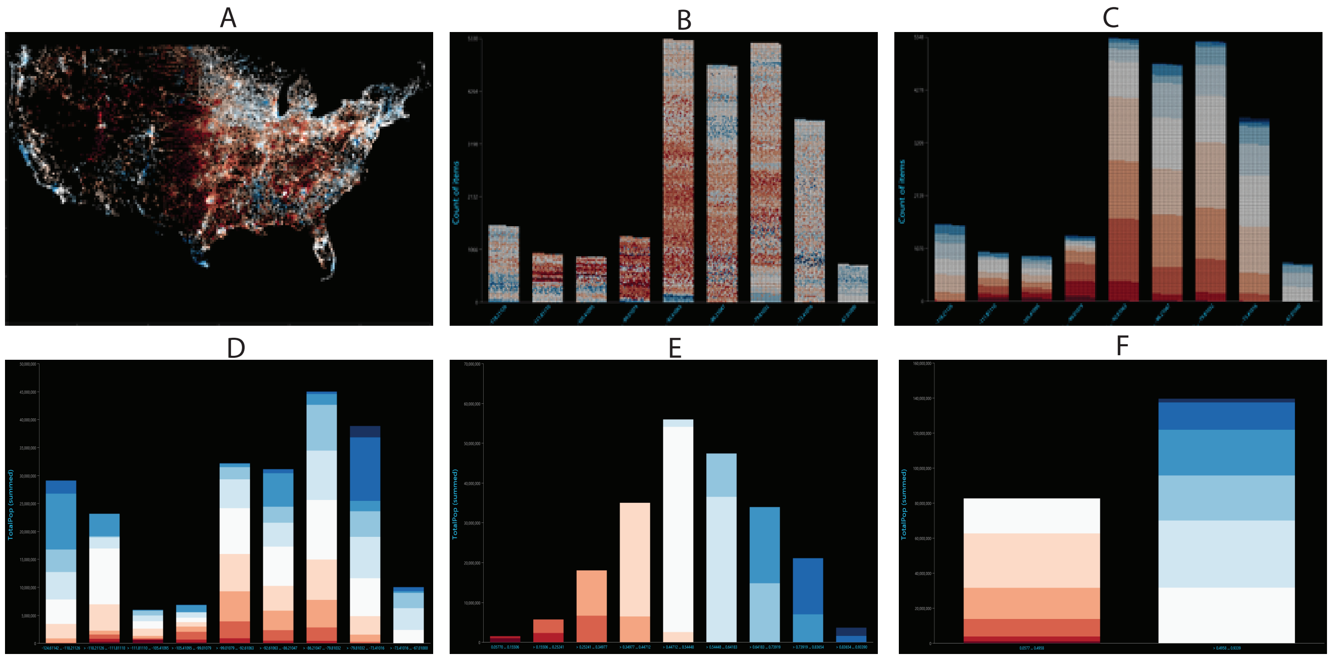


Fig. 12. A sequence exploring the 2010 election. A: Scatter Plot Colored by Voting Percent, B: Binned by Longitude, C: Sorted by Voting Percent, D: Summed by Total Population, E: Binned by Voting Percent, F: Changed to 2 Bins.

can track the movements of all units as they move. However the transitions reinforce that the different views are just different ways of laying out all the units and that each representation is related because the units are always associated with the same row. Anecdotally, we've observed that users have more difficulty in understanding the meaning of sequences of visualizations when without animated transitions.

4.3 Unit Representations

Units can be represented as shapes, text, glyphs, or even as tiles of images (see figure 11). Improved performance for images is accomplished by computing mip-maps for the images [34] and using the appropriate texture as needed.

4.4 Deployment and Initial Testing

The prototype was deployed both internally to our company via the web and externally to partners focused beta-testing new ideas. External partners used a plug-in to Excel so that SandDance could be opened directly within Excel and have direct access to the sheet, thus allowing the data to be manipulated in Excel. Selections and filters were communicated back to Excel so that state could be resumed between sessions. The plugin was made available to over 50 people who used the SandDance system after watching some introductory videos. In order to prompt more complete feedback, we challenged 20 users with progressively harder questions over the course of a week. We designed our questions to cover a wide array of typical analytic tasks (shown below) that were outlined in Amar & Stasko [2].

Table 1. Visual Analytic Task Categories

Filter	Characterize Distribution	Cluster
Determine Range	Computer Derived Value	Correlate
Find Extremum	Sort	Find Anomalies

Sampling of Questions asked:

1. Using the Adventure Works Sales data set, how many total bikes were sold? (Compute Derived Value)
2. Using the Adventure Works Sales data set, which month had the most total items sold? (Find extremum)
3. Using the Census data set, how many CDPs are in Washington State? Florida State? (Determine Range)
4. In the US, how many CDPs contain "King" in their county name?(Find)
5. A common expression for disasters is 'women and children first'. Was this followed on the titanic? (Characterize Distribution)
6. In what cabin class were most of the younger children located? How many survived and how many died in that class? (Cluster)
7. What is the relationship between income and education in the Census data set? (Correlate)

All 20 people attempting the task were successful (though since they did the work on the own time, outside of a lab, we did not log task completion times). People liked the quiz both as a gentle introduction to the tool, as well as revealing the power of SandDance as an analysis tool. It was interesting to note the different strategies that people used to find answers including general search, search based on a single unit, sorting, changing visualization representation, visualization, and filter out the results and change the visualization type again, etc.

Overall, feedback from the deployment, as well as over 200 hours of one-on-one demonstrations has been extremely encouraging. In particular, teachers have requested the use of SandDance in classrooms both to help illustrate data stories as well as to give a better understanding of the elements of data exploration. Many people comment that they like the way that every row of the data is represented and how the movements (like grains of sand) help indicate overall structure in the data. They liked how the the particles show the users a visual sense (evidence) of the actual numbers behind the aggregate values' and how engaging and even playful it became to explore the data.

Many improvements were suggested; several of these are discussed in the future work section.

5 Future Work

There is a great deal of future work in this space. While unit visualizations themselves have been shown to be useful, there are many instances where additional information, or layers of representations would be helpful. This includes an underlying map for geographic visualizations, or underlying annotations such as median lines, labels, counts, quartiles, or annotations.

A hybrid model that moves between conventional aggregations and unit visualization could help a system scale to deal with larger datasets and be better suited to client-server communication. Once the data has been filtered to an appropriate level, the unit representations can replace the aggregates. Appropriate UI metaphors for dealing with hybrid representations will be interesting to explore.

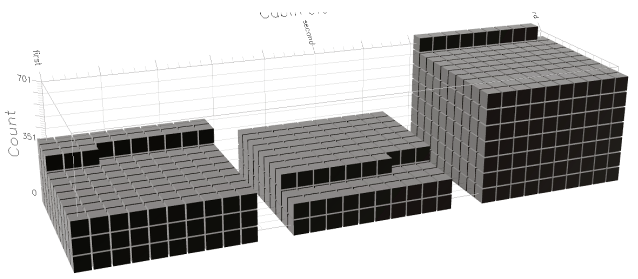


Fig. 13. Framework can be extended to 3D for unit visualizations.

We currently record the entire history of actions and allow users to step forwards or backwards through their interaction history. The text file of these interactions can be manually edited to produce a tour, but a more useful tour interface would be desirable since it has been found that telling stories with the data is one of the more compelling aspects of unit visualizations.

The framework can easily be generalized to more dimensions. Containers can be rectangular prisms instead of rectangles and units can in turn be 3D objects 13. While the utility of 3D itself is often hotly debated, it will be interesting to explore how the addition of another dimension will change the overall usefulness of the system.

Transitions right now are fairly straightforward, but more semantically meaningful transitions, such as bundling trajectories [10], or ScatterDice [11] like transitions between representations could be helpful.

Finally, all the data from deployment to over 50 people has been logged, and it will be interesting to see what common patterns of behavior were used, especially when addressing some of the questions asked in the initial deployment testing.

We intend to release SandDance to a wider audience to collect more feedback on patterns of use in visual analysis and storytelling

6 Conclusion

Unit Visualizations are not only historically interesting artifacts, but also useful representations for current interactive exploratory and presentation systems with data. We have discussed a framework that helps define the space of unit visualizations and demonstrated a system that uses unit visualizations at its core. It excels as ways of slicing and dicing the data through a wide number of selection types as well as comprehensible transformations between different views of multi-dimensional data.

References

[1] C. Ahlberg. Spotfire: an information exploration environment. *ACM SIGMOD Record*, 25(4):25–29, 1996.

[2] R. Amar and J. Stasko. A knowledge task-based framework for design and evaluation of information visualizations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 143–150. IEEE, 2004.

[3] S. Amershi, M. Chickering, S. Drucker, Lee, P. B. Simard, and J. Suh. Modeltracker: Redesigning performane analysis tools for machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2015.

[4] B. B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 71–80. ACM, 2001.

[5] C. R. Blyth. On simpson's paradox and the sure-thing principle. *Journal of the American Statistical Association*, 67(338):364–366, 1972.

[6] F. Chevalier, P. Dragicevic, and C. Hurter. Histomages: fully synchronized views for image editing. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 281–286. ACM, 2012.

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

[8] P. Dragicevic, A. Bezerianos, W. Javed, N. Elmqvist, and J.-D. Fekete. Temporal distortion for animated transitions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2009–2018. ACM, 2011.

[9] S. M. Drucker, C. Wong, A. Roseway, S. Glenner, and S. De Mar. Mediabrowser: reclaiming the shoebox. In *Proceedings of the working conference on Advanced visual interfaces*, pages 433–436. ACM, 2004.

[10] F. Du, N. Cao, J. Zhao, and Y.-R. Lin. Trajectory bundling for animated transitions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2015.

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

[12] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *Visualization and Computer Graphics, IEEE Transactions on*, 16(3):439–454, 2010.

[13] S. Few. Unit charts are for kids., 2010.

[14] M. Friendly and D. Denis. Published online in wiley interscience (www.interscience.wiley.com). doi 10.1002 /jhbs.20078 2005 wiley periodicals, inc. the early origins and development of the scatterplot.

[15] J. Heer and G. G. Robertson. Animated transitions in statistical data graphics. In *In IEEE Information Visualization (InfoVis*, 2007.

[16] H. Hofmann. Exploring categorical data: interactive mosaic plots. *Metrika*, 51(1):11–26, 2000.

[17] J. Hullman and N. Diakopoulos. Visualization rhetoric: Framing effects in narrative visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2231–2240, 2011.

[18] J. Hullman, S. Drucker, N. H. Riche, B. Lee, D. Fisher, and E. Adar. A deeper understanding of sequence in narrative visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2406–2415, 2013.

[19] S. Huron, Y. Jansen, and S. Carpendale. Constructing visual representations: Investigating the use of tangible tokens. 2014.

[20] S. Huron, R. Vuillemot, and J.-D. Fekete. Visual sedimentation. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2446–2455, 2013.

[21] D. A. Keim, M. C. Hao, U. Dayal, and M. Hsu. Pixel bar charts: a visualization technique for very large multi-attribute data sets. *Information Visualization*, 1:20–34, 2002.

[22] O. Neurath. *International Picture Language; the First Rules of Isotype: With Isotype Pictures*. K. Paul, Trench, Trubner & Company, 1936.

[23] C. Petzold. *Applications= code+ markup: a guide to the Microsoft Windows presentation foundation*. Microsoft Press Redmond, WA, 2006.

[24] P. Rudman. *How mathematics happened: the first 50,000 years*. Prometheus Books, 2007.

[25] J. M. Rzeszotarski and A. Kittur. Touchviz:(multi) touching multivariate data. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1779–1784. ACM, 2013.

[26] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1139–1148, Nov 2010.

[27] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional databases. *Communications of the ACM*, 51(11):75–84, 2008.

[28] E. R. Tufte and P. Graves-Morris. *The visual display of quantitative information*, volume 2. Graphics press Cheshire, CT, 1983.

[29] J. W. Tukey. *Exploratory Data Analysis*. Behavioral Science: Quantitative Methods. Addison-Wesley, Reading, Mass., 1977.

[30] H. Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010.

[31] H. Wickham and H. Hofmann. Product plots. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2223–2230, 2011.

[32] L. Wilkinson. Dot plots. *The American Statistician*, 53(3):276–281, August 1999.

[33] L. Wilkinson. *The Grammar of Graphics*. Springer-Verlag New York, Inc., New York, NY, USA, 1999.

[34] L. Williams. Pyramidal parametrics. In *ACM Siggraph Computer Graphics*, volume 17, pages 1–11. ACM, 1983.

[35] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information Visualization*, 4(4):239–256, 2005.