Gatherplots: Extended Scatterplots for Categorical Data

Deokgun Park, Sung-Hee Kim, and Niklas Elmqvist, Member, IEEE

Abstract—Scatterplots are a common tool for exploring multidimensional datasets, especially in the form of scatterplot matrices (SPLOMs). However, scatterplots suffer from overplotting when categorical variables are mapped to one or two axes, or the same continuous variables are used for both axes. Previous methods such as histograms or violin plots for these cases aggregate marks, which makes brushing and linking difficult. To improve this, we propose gatherplots, an extension of scatterplots to manage overplotting for categorical data, while keeping individual object identities. In gatherplots, every data point that maps to the same position coalesces to form a stacked entity, thereby making it easier to see the overview of data groupings. The size and aspect ratio of data points can also be changed dynamically to make it easier to compare the composition of different groups. In the case of a categorical variable vs. a categorical variable, we propose a heuristic to decide bin sizes for optimal space usage. This means that make better use of visual space to show the overall distribution. To validate our work, we conducted a crowdsourced user study that shows that gatherplots enable users to judge the relative portion of subgroups more quickly and more correctly than when using jittered scatterplots.

Index Terms—Scatterplots; overplotting; scatterplot matrices (SPLOMs).

1 INTRODUCTION

Scatterplots—one of the most widely used types of statistical graphics [1], [2], [3]—are commonly used to visualize two continuous variables using visual marks mapped to a two-dimensional Cartesian space, where the color, size, and shape of the marks can represent additional dimensions. It can also be used for exploring multidimensional datasets in the form of scatterplot matrices (SPLOM), where all the possible combinations of axes are presented in table form. However, scatterplots are so-called overlapping visualizations [4] in that the visual marks representing individual data points may begin to overlap each other in screen space in situations when the marks are large, when there is insufficient screen space to fit all the data at the desired resolution, or simply when several data points share the same value. In fact, realistic multidimensional datasets often contain categorical variables, such as nominal variables or discrete data dimensions with a small domain, which lead to many data points being mapped to the exact same screen position. This kind of overlap is known as overplotting (or overdrawing) in visualization, and is problematic because it may lead to data points being entirely hidden by other points, which in turn may lead to the viewer making incorrect assessments of the data. As can be seen in Figure 1, there are three situations for mapping variables to axes in scatterplots when overplotting is inevitable:

- Plotting categorical vs. continuous variables gives rise to points forming line patterns ((b) and (c) in Figure 1);
- Plotting categorical vs. categorical variables gives rise to points forming single dot patterns (Figure 1(d)); and
- Plotting the same continuous variable on both axes gives rise to points forming diagonal line patterns (Figure 1(a)).

![Fig. 1. A scatterplot matrix for a car dataset with one continuous variable MPG and one categorical variable Cylinders showing limitations of scatterplots when managing categorical variables. In (a), a scatterplot with the same variable for both axes results in a diagonal line. In (b) and (c), a scatterplot with a continuous vs. a categorical variable results in horizontal or vertical line patterns. In (d), a scatterplot with two categorical variables results in a dot pattern.](image-url)
Several approaches have been proposed to address this problem [5], the most prominent being transparency, jittering, and clustering techniques. The first, changing transparency, does not so much address the problem as sidestep it by making the visual marks semi-transparent so that an accumulation of overlapping points are still visible. However, this does not scale for large datasets, and also causes blending issues if color is used to encode additional variables. Jittering perturbs visual marks using a random displacement [6] so that no mark falls on the exact same screen location as any other mark, but this approach is still prone to overplotting for large data. It also introduces uncertainty that is not aptly communicated by the scatterplot since marks will no longer be placed at their true location on the Cartesian space. Other approaches still attempt to organize overlapping marks into visual groups that summarize their distribution, such as histograms, violin plots, and kernel density estimation (KDE) plots [7], [8], [9]. However, this comes at the cost of losing the identity of individual points, which can be problematic when filtering or searching; e.g., brushing data points is difficult in histograms [9].

In this paper, we propose the concept of gathering as an alternative to scattering and jittering, and then show how we can use this visual transformation to define a novel visualization technique called a gatherplot. Gathering is a generalization of the linear mapping used by scatterplots, and works by first partitioning the graphical axis into segments based on the data dimension and then organizing points into stacked groups for each segment to avoid overplotting. This means that the gather operation relaxes the continuous spatial mapping commonly used for a graphical axis; instead, each discrete segment occupies a certain amount of screen space that all maps to the same data value. This is communicated using graphical brackets on the axis that shows the value or interval for each segment (Figure 2(b)). Beyond the gatherplot technique, we also show how the gather transformation can be embedded into a Magic Lens [10] for use in a normal scatterplot.

The contributions of our paper are the following: (1) the concept of the gather visual transformation as a generalization of linear visual mappings; (2) the gatherplot technique, an application of the gather operation to scatterplots to solve the overplotting problem; (3) the GatherLens interaction technique where gathering is applied on a local level; and (4) results from a crowdsourced graphical perception on the effectiveness of different modes of gatherplots. Next we first review the literature on statistical graphics and overplotting. We then present the gather operation and use it to define gatherplots. This is followed by our crowdsourced evaluation. We finally describe the GatherLens technique and close with conclusions, and our future plans.

2 BACKGROUND

Our goal with gatherplots is to generalize scatterplots to a representation that maintains its simplicity and familiarity while eliminating overplotting. With this in mind, below we review prior art on mitigating overplotting using appearance or distortion. We also discuss visualization techniques specifically designed for categorical variables.

2.1 Characterizing Overplotting

While there are many ways to categorize visualization, Fekete and Plaisant [4] introduced a classification particularly useful for our purposes that splits techniques into two types:

- **Overlapping visualizations**: No layout restrictions on visual marks is enforced, leading to overplotting. (Scatterplots, node-link diagrams, parallel coordinate plots.)
- **Space-filling visualizations**: Layouts that fill the available space to avoid overlap. (Treemaps, matrices, maps.)

Fekete and Plaisant [4] investigated the overplotting phenomenon for a 2D scatterplot, and found that it has a significant impact as datasets grow. The problem stems from the fact that even with two continuous variables that do not share any coordinate pairs, the size ratio between the visual marks and the display remains more or less constant. Furthermore, most datasets are not uniformly distributed. This all means that overplotting is bound to happen for realistic datasets.

Ellis and Dix [5] survey the literature and derive a general approach to reduce clutter. According to their treatment, there are three ways to reduce clutter in a visualization: by changing the visual appearance, by distorting visual space, or by presenting data over time. Some trivial but impractical mechanisms they list include decreasing mark size, increasing display space, or animating the data. Below we review practical approaches based on appearance and distortion.

2.2 Appearance-based Methods

Practical appearance-based approaches to mitigate overplotting include transparency, sampling, kernel density estimation (KDE), and aggregation. Transparency changes the opacity of the visual marks, and has been shown to convey overlap for up to five occurrences [11]. However, there is still an upper limit for how much overlap is perceptible to the user, and the blending caused by overlapping marks of different colors makes identifying colors difficult.

Sampling uses stochastic methods to statistically reduce the data size to visualize [12]. This may reduce the amount of overplotting, but since the sampling is random, it cannot be reliably eliminated. Furthermore, one of the core strengths of a scatterplot is its ability to show outliers effectively, whereas sampling will likely eliminate all outliers.

KDE [13] and other binned aggregation methods [7], [8], [9], [14] replace a cluster of marks with a single entity that has a distinct visual representation. Splatterplots [8] overcome this by combining individual marks with aggregated entities, using marks to show outliers and aggregated entities to show the general trends. While this technique is effective against overplotting for continuous variables, it was not designed to handle categorical ones. Therefore, the pioneering generalized plot matrices (GPLOMs) [9] were proposed to solve this particular problem by adopting non-homogeneous plots into a matrix. The technique uses a histogram for categorical vs. continuous variables, and a
2.3 Distortion-based Methods

Distortion-based techniques avoid overplotting by changing the spatial mapping of the space and have the advantage that it keeps the identity of individual data points. The canonical distortion technique is jittering, where a random displacement is used to subtly modify the exact screen space position of a data point. This has the effect of spreading data points apart so that they are easier to distinguish. However, most naive jittering mechanisms apply the displacement indiscriminately to all data points, regardless of whether they are overlapping or not. This has the drawback of distorting points away from their true location on the visual canvas, and still does not completely eliminate overplotting.

Bézieranos et al. [15] use a more structured approach to displacement, where overlapping marks are organized onto the perimeter of a circle. The circle is grown to a radius so that all marks fit, which means that its size is also an indication of the number of participating points. However, this mechanism still introduces uncertainty in the spatial mapping, and it is also not clear how well it scales for very dense data. Nevertheless, it is a good example of how deterministic displacement can be used to great effect for eliminating overplotting.

Trutschl et al. [6] propose a deterministic displacement (“smart jittering”) that adds meaning to the location of jittering based on clustering results. Similarly, Shneiderman et al. [16] propose a related structured displacement approach called hieraxes, which combines hierarchical browsing with two-dimensional scatterplots. In hieraxes, a two-dimensional visual space is subdivided into rectangular segments for different categories in the data, and points are then coalesced into stacked groups inside the different segments. This work inspired gatherplots, which refines the layout and design of hieraxes further.

2.4 Visualizing Categorical Variables

While we have already ascertained that scatterplots are not optimal for categorical variables, there exists a multitude of visualization techniques that are (e.g. [17], [18], [19]). Simplest among them are histograms, which allows for visualizing the item count for each categorical value [20]. Boxplots and violin plots show the distribution of continuous variables over categorical variables [21]. While hieraxes, histograms, and treemaps show the distribution of continuous vs. categorical variables. One way is to apply binning to continuous variables to create groups of values. However, the optimal number of bin depends on statistical characteristics of the data and the required task. Dot plots by Wilkinson [22] renders continuous univariate variables without overplotting by stacking nodes within dot size. Dang et al. [23] extended this to scatterplots by stacking nodes whose values are similar in 3D visual space. These pioneering works provide the theoretical background for the determination of optimal bin size for gatherplots.

Another method for visualizing categorical data that is of practical interest is for making inferences based on statistical and probabilistic data. Cosmides and Toody [24] used frequency grids as discrete countable objects, and Micallef et al. [25] extend this with six different area-proportional representations of categorical data organized into different classes. Huron et al. [26] suggested using sedimentation as metaphor where individual objects coming from a data stream gradually transforms into aggregated areas, or strata.

3 The Gather Transformation

Position along a common scale is the most salient of all visual variables [27], [28], and so mapping a data dimension to positions on a graphical axis is a standard operation in data visualization. We call this mapping a visual transformation. However, most statistical treatments of data, such as Stevens’ classical theory on the scale of measurements [20], do not take the physical properties of display space into account. This is our purpose in the following section.
3.1 Problem Definition

Let $V = \langle f, s \rangle$ be a visual transformation that consists of a transformation function $f$ and a mark size $s$ (pixels). Furthermore, assume that $f$ transforms a data point $p_d \in D$ from a data dimension $D$ to a coordinate on a graphical axis $p_c \in C$ by $f(p_d) = p_c$. Given a dataset $D_1 \subseteq D$, we say that a particular visual transformation $V_f$ exhibits overlap if

$$\exists p_x, p_y \in D_1 \land x \neq y : |f_j(p_x) - f_j(p_y)| < s_j.$$ 

In other words, overlap occurs for a particular dataset and visual transformation if there exists at least one case where the visual marks of two separate data points in the dataset fall within the same interval on the graphical axis. The overlap index of a dataset and visual transformation is defined as the number of unique pairs of points that overlap. For a one-dimensional visualization, only a single transformation is used and the visualization and dataset is said to exhibit overplotting if it exhibits overlap. For a two-dimensional visualization, however, the visualization and dataset will only exhibit overplotting if there is overlap in both visual transformations and data dimensions. Analogously, the overplotting index is the unique number of overplotting incidences for that particular visual transformation and dataset.

This has two practical implications: (1) even a dataset that consists only of nominal variables may not exhibit overplotting if there is only at most one instance of each nominal value, and (2) a dataset consisting of continuous values may still exhibit overplotting if any two points in the dataset are close enough that they get mapped to within the size of the visual marks on the screen. The corollary is basically that overplotting is a function of both visualization technique and dataset.

3.2 Definition: The Gather Transformation

We build on the previous idea of structured displacement [15], [16] by proposing a novel visual transformation function called a gather transformation $f_{\text{gather}}$ that nonlinearly segments the graphical axis $C$ and organizes data points in each segment to eliminate overplotting.

The gather transformation $V_{\text{gather}} = \langle f_{\text{gather}}, s_{\text{gather}} \rangle$ consists of a transformation function $f_{\text{gather}}$ that maps data points $p_d \in D$ to coordinates $p_c \in C$, and a visual mark sizing function (instead of a scalar) $s_{\text{gather}}$ that yields a visual mark size given the same data point. The gather transformation function is special in that it eliminates overplotting by subdividing the graphical axis $C$ into $n$ contiguous segments $C = \{C_1, C_2, \ldots, C_n\}$, where $n$ is the size of the domain of the gather transformation function, i.e., the number of unique elements in the data dimension $D$. When mapping a data point $p_d$ to the graphical axis, $f_{\text{gather}}$ will return an arbitrary graphical coordinate $p_c \in C_i$ for whatever coordinate segment $C_i$ that $p_d$ belongs to.

Practically speaking, coordinates $p_c \in C_i$ will be chosen to efficiently pack visual marks into the available display space without causing overplotting (i.e., using a regular spacing of size $s_{\text{gather}}$). Several different methods exist for adapting the gather transformation to the dataset $D$. One approach is to keep the segments $C_1, \ldots, C_n$ of equal size and find a constant visual mark size $s_{\text{gather}}(p_d) = s_{\text{max}}$ that ensures that all points fit within the most dense segment. The constant mark size makes visual comparison straightforward. Another approach is to adapt segment size to the density of the data while still keeping the mark size constant. This will minimize empty space in the visual transformation and allows for maximizing mark size. A third approach is to vary mark size proportionally to the number of points in a segment. This will make comparison of the absolute number of points in each segment difficult, but may facilitate relative comparisons if marks are distinguished in some other way (e.g., using color).

For data dimensions $D$ that have a very large number of unique values, it often makes sense to first quantize the data using a function $p_q = Q(p_d)$ so that the number of elements $n$ is kept manageable (on the order of 10 or less for most visualizations). For example, a data dimension representing a person’s age might heuristically be quantized into ranges of 10 years: 0-9 years, 10-19 years, 20-29 years, and so on.

In a gather transformation, the coordinate axis has been partitioned into segments, where the order of segments on the axis depends on the data. For nominal data, the segments can be reordered freely, both by the algorithm...
and by the user. For ordinal or quantized data, the order is given by the data relation. Furthermore, it often makes sense to be able to order points inside each segment $C_i$ using the gathering transformation function $f_{\text{gather}}$, for example using a second data dimension (possibly visualized using color) to group related items together.

Appropriate visual representations of data where the gather transformation has been applied are also important. The stacked entities of gathered points—one per coordinate segment $C_i$—should typically maintain object identity, so that each constituent point and their size is discernible as a discrete visual mark. Similarly, a visual representation of the segmented graphical axis should externalize the segments as labeled intervals instead of labeled major and minor ticks; this will also communicate the discontinuous nature of the axis itself to the viewer.

### 3.3 Using the Gather Transformation

To give an example in one-dimensional space, parallel coordinate plots [29] use multiple graphical axes, one per dimension $D_i$, and organize them in parallel while rendering data points as polylines connecting data values on one axis to adjacent ones. However, traditional parallel coordinate plots merely use a scatter transformation on each graphical axis, which makes the technique prone to overplotting. Multiple authors have studied ways of mitigating this problem, for example by reorganizing the position of nominal values [30], using transparency, applying jitter, or by clustering the data [7].

However, an alternative approach is to use the gather representation for each graphical axis to minimize overplotting. This will cause each axis to be segmented into intervals, and we can then resize segments according to the number of items falling into each segment so that segments with many data points become proportionally larger than those with fewer points. Finally, if the data dimensions represent nominal data, it may make sense to use a global segment ordering function so that there is a minimum of lateral movement for the majority of points as they connect to adjacent axes. This will also minimize line crossings between the parallel axes. This particular visualization technique—a parallel coordinate plot with the gather transformation applied to each graphical axis—is essentially equivalent to parallel sets [19].

In fact, by applying our generalized gather transformation to the axis, we are actually proposing a new type of stacked visualization where each entity is still represented by lines. In a sense, this technique combines parallel coordinates and parallel sets because the grouped lines maintain the illusion of a single entity for an axis with nominal categorical values (similar to parallel sets), yet integrates directly with a parallel coordinate axis with continuous values. The main difference is that the new parallel coordinate/set variation allows each axis to be either categorical or continuous, meaning that one axis can represent the gender and the next can represent the height of person.

### 4 Gatherplots: A 2D Gathering Representation

Applying gathering to two perpendicular axes defining a Cartesian space results in a gatherplot: a 2D distortion-based extension of scatterplots that gathers data points into stacked groups, thereby eliminating overplotting without losing the identity of individual data points. Compared to jittering, which relies on random permutation, gathering organizes visual marks according to visual features, so that the resulting group of objects forms a meta-object. According to Haroz and Whitney [31], grouping marks by feature helps in performing perceptual tasks such as finding outliers, counting items, seeing trends, and so on. The technique is particularly designed for visualizing categorical variables. Below we discuss the open design parameters for the technique, including layout, aspect ratio, and item shapes.

#### 4.1 Layout

Gatherplots eliminate overplotting by gathering marks with similar visual properties into stacked groups. This is inspired by previous works such as hieraxes [16] or frequency grids [24], [25]. However, there are many design possibilities for organizing the visual representation depending on the context, especially on the size distribution of each group, the aspect ratio of assigned space, and the task at hand. As a result, we derive the following three layout modes (see Figure 3):

- **Absolute mode**: Here stacked groups are sized to follow the aspect-ratio of the assigned region. The size of the items are determined by the maximum length dots which can fill the assigned region without overlapping. This means with the same assigned space, the groups with the maximum number of members determines the overall size of the nodes (Figure 3(a)).

- **Normalized mode**: In this mode, the mark size and aspect ratio is adapted so that every stacked group has equal dimensions. This makes it easier to investigate ratios when the user is interested in the relative distributions of subgroups rather than the absolute number of members. Items also change their shape from a circle (absolute mode) to a rounded rectangle (Figure 3(b)). Normalized mode is useful for two specific tasks:
  - Finding the ratio of the subgroups in a group (Figure 3). Because groups of different size are normalized to the same geometric area, any comparison results in a relative comparison, which can aid statistical Bayesian reasoning [25].
  - Finding the distribution of outliers. When there are many items on the screen for absolute mode, all marks must be reduced in size. This can make outliers hard to locate. When normalized mode is used, the outliers are expanded to fill the assigned space, making them easier to see.

- **Streamgraph mode**: Here stacked groups are reorganized so that they maintain the same number of
elements in their shorter edge. This mode is used for regions where the ratio of width and height are drastically different (in our prototype implementation, we use a heuristic threshold aspect ratio value of 3 for activating this mode). This means there are usually many times more groups in the axis in parallel with shorter edges. A good example is for visualizing the population distribution with regards to gender and age; the resulting gatherplot approaches ThemeRiver [32] as the number of entities increases (Figure 3(c)).

The choice between absolute and streamgraph mode happens automatically based on the aspect ratio of assigned space and without the need for user intervention. Therefore, only a simple interaction is required to toggle between absolute and normalized mode. However, we do expose a setting to manually toggle between these modes as well.

4.2 Managing Continuous Variables

To use gatherplots for continuous variables, we apply binning to partition the variable into discrete intervals. The resulting visualization resembles dots plots by Wilkinson [22], where bin size is equal to dot size. The size of individual bins is important for binning because it determines the spatial accuracy and legibility of the visualization.

Wilkinson proposed $\frac{1}{2}5n^{-1/2}$ as the optimal dot size for dot plots. This creates reasonable dot plots for fixed aspect ratio of 5 to 1, which is common in statistical charts assuming normal distribution of nodes. However, gatherplot requires two different assumptions: First, the aspect ratio varies according to the space given to the categorical variables. Second, the dot size or bin size is determined by the global maximum in the dataset, which may not be in the same cluster. Furthermore, because bin size is the same as dot size, selecting bin size can be thought of as a trade-off between accuracy and legibility. Using very small bin size and dot size increases the spatial accuracy, but results in poor legibility, and vice versa. Balancing accuracy vs. legibility is common in visualization for large datasets; for example, splatterplots limit the information shown to users based on the available visual space [8]. Similarly, gatherplots choose bin size based on spatial accuracy and legibility. When the visual space is small, we use a comparably large bin size to increase dot size, thus resulting in poor spatial accuracy and high legibility; for larger space allocations, the bins can be made smaller to increase accuracy without loss of legibility. This is shown in the Figure 4.

Figure 5 (a) shows how gatherplots handle the situation when continuous variables are assigned to both axes, causing both to be binned. The plot is using normalized mode with two random variables. The normalized mode makes it easier to identify the outliers and the distribution of outliers. Furthermore, the case of scatterplots with the same continuous variables on both axes can be treated as a special case of continuous vs. categorical variables. Here, the gatherplot is rotated to maintain integrity with scatterplots (Figure 2 (c)).

One limitation of gatherplots is that the technique requires binning to manage a continuous variable, yet binning creates arbitrary boundaries that can be misleading. However, using both gatherplots and scatterplots in different views makes this problem less severe because the analyst can simply choose the visual representation most suited for a particular task.

4.3 Undefined Axis Mapping

Scatterplots have traditionally been used to view correlations between two variables. However, for a multidimensional exploration task, one subtle difficulty is when the user wants to see only the effect of a single variable, without definition on the other axes. In gatherplots, the logical extension of an undefined axis is the aggregation of all nodes in a single group along that axis. Figure 6 shows an example of this using a dataset on survivors of the Titanic.

Survivors of Titanic

---

**Fig. 4.** Choosing optimal bin size based on available display space. In (a), there is enough space so that the dot size can be maximized, improving spatial accuracy. In comparison, in (b) the assigned space is small, so the dot size is determined so that the most crowded bin interval will fit within the assigned space. This results in two different overviews even though the two plots have identical aspect ratio.

**Fig. 6.** Gatherplot showing survivors of the Titanic. (a) All people on board. Note that the X and Y axis are not defined. (b) Survivors with color coding. (c) Distribution of survivors over class variables. Here the Y axis is undefined. (d) Distribution of survivors over the gender. Here the X axis is undefined.
4.4 Visual Design

Gatherplots build on the same visual language as scatterplots. However, some aspects are different; below we discuss our design choices for visual mark shape as well as tick marks.

4.4.1 Visual Marks

Scatterplots typically use a small circle or dot as a visual representation for items, but many variations exist that use glyph shapes to convey multidimensional variables [1, 33], [34], [35], [36]. However, in normalized mode, sometimes the aspect ratio of visual marks changes according to the aspect ratio of the space assigned to that value. Also, as gathering changes the size of marks to fit in one cluster, sometimes the marks size becomes too small or too large compared to other marks. This results in several unique design considerations for item shapes.

Based on our experience with several alternate designs, we recommend using a rectangle with constant rounded edge without using stroke lines. Using constant rounded edge allows the nodes to be circular when the mark is small, as in Figure 3(b), and a rectangle to show the degree of stretching, as shown in Figure 3(b). As for avoiding stroke lines, such lines become dominant when nodes shrink below a certain size. Eliminating them entirely curtails this problem.

4.4.2 Interval Tick Marks

Because we are representing ranges rather than single points, the single line type tick marks for scatterplots are not appropriate for gatherplots; instead, ticks should communicate the partitioned segments on the axes. Without this visual representation, when the user is confronted with a number, it can be confusing to determine whether adjacent nodes with different offset has same value or not.

After considering a few visual design alternatives, we recommend a bracket type marker for this purpose. Figure 7 shows design alternatives of tick mark for representing ranges. The bracket is optimal in that it uses minimal ink and creates less density with adjacent ticks.

4.5 Interaction

Gatherplots support the same types of interactions as scatterplots. However, some additional interaction techniques are required to specifically control the gathering transformation.

For example, when exploring multidimensional datasets, it is crucial to have a mechanism to filter unwanted data. To support this process in gatherplots, we provide an optional mechanism to go back to the original continuous linear scale function. We allow each axis tick have an interactive control mechanism to go back to the original continuous linear scale.

5 Gatherplots: Implementation

We have implemented a web-based demonstration of gatherplots using D3.js1 and Angular.js2. The prototype can be accessed online at http://www.gatherplot.org, and allows users to load various datasets into a gatherplot. The visualization can be compared to scatterplots and jittered scatterplots with a single click. In the top right area, an

---

interactive guide is provided where users can follow step-by-step instructions in a guided tour of gatherplots.

6 GATHERPLOTS: EVALUATION

This study was designed to demonstrate the effectiveness of gatherplots, in particular its different layout modes with categorical vs. categorical variables. Crowdsourcing platforms have been widely used and have shown to be reliable platforms for evaluation studies [37], [38]. Therefore, we conducted our experiment on Amazon Mechanical Turk.3 This also gave us the opportunity to study the utility of the technique for the general population, who do not have specific statistical training.

6.1 Experimental Design

We selected jittered scatterplots as the baseline condition, as this technique is a widely accepted standard technique maintaining consistency with scatterplots. We also wanted to measure the efficiency of different modes of gatherplots. Therefore, we designed the experiment to have four conditions: scatterplots with jittering (jitter), gatherplots with absolute mode (absolute), gatherplots with normalized mode (normalized), and gatherplots with a toggle to switch between absolute and normalized mode (both). We adopted a between-subjects design to eliminate learning effects from experiencing other modes.

6.2 Participants

A total of 240 participants (103 female) completed our survey. Because some questions asked about concepts of absolute numbers and probability, we limited demographic to the United States to remove the influence of language. To ensure the quality of the workers, the qualification of workers were the approval rate of more than 0.95 with number of hits approved to be more than 1,000. Only three of 240 participants did not use English as their first language. 119 people had more than bachelor’s degree, with 42 people having high school degree. We filtered random clickers by removing any trials where the completion time was shorter than a reasonable time (5 seconds). This yielded a total of 211 participants.

6.3 Task

As scatterplots can support many types of tasks, it is difficult to come up with a representative task. In the end, we selected retrieving a value as a low-level task, and comparing and ranking as a high-level task. For the comparing and ranking task, two different types of questions were asked: the tasks to consider absolute values such as frequency and tasks that consider relative values such as percentage. Therefore, for one visualization, 5 different questions were generated. For gatherplots, our interest is in the difference between task considering absolute values and relative values. The five tasks are as follows:

- T1: retrieve value considering one subgroup.
- T2: comparing absolute size of subgroup between groups.
- T3: ranking absolute size of subgroup between groups.
- T4: comparing relative size of subgroup between groups.
- T5: ranking relative size of subgroup between groups.

To reduce the chance of one chart being optimal by luck for a specific task, two charts of same problem structure were provided. This yielded a total of 10 questions for each participant. Each question was followed by the question asking confidence of estimation with a 7-point Likert scale, and the time spent for each question was measured.

6.4 Dataset

We used a dataset about the Survivors of Titanic. Each of the 2,201 survivors had four dimensions, which were all categorical variables: class (4 levels), sex (2 levels), port of entry (4 levels), survival status (2 levels). The five tasks above were asked for two views with different dimensions. One view visualized class on X-axis, sex on Y-axis, and survival using color. The second view visualized survived on the X-axis, class on the Y-axis, and port of entry using color.

6.5 Hypotheses

We believe that different types of tasks will favor from different type of layouts. Therefore our hypotheses are as follows:

H1: For retrieving value considering one subgroup (T1), both absolute and normalized modes yield better accuracy than jitter mode.

3. https://www.mturk.com
H2 For tasks considering absolute values (T2 and T3), absolute mode yields best accuracy over other modes.

H3 For tasks considering relative values (T4 and T5), normalized mode yields best accuracy over other modes.

6.6 Results

The results were analyzed with respect to the accuracy (correct or incorrect), time spent, and confidence of estimation. Based on our hypotheses, we analyzed the different modes of layout for each type of question: retrieve value, absolute value task, and relative value task.

6.6.1 Accuracy

The number and percentage of participants who answered correct and incorrect answers are shown in Figure 9. In total, we recruited 42 participants for jitter, 56 participants for absolute, 56 participants for normalized, and 57 participants for interactive mode.

As the measure for each question was either correct or incorrect, a logistic regression was employed. For the retrieving-value task (T1), both the absolute mode and normalized mode had significant main effects (Wald Chi-Square = 18.58, p < 0.01, Wald Chi-Square = 21.05, p < 0.01, respectively) with a significant interaction effect (Wald Chi-Square = 19.53, p = 0.03) (H1 confirmed). For absolute-value tasks (T2 and T3), both the absolute mode and normalized mode had significant main effects (Wald Chi-Square = 10.35, p < 0.01, Wald Chi-Square = 10.35, p < 0.01, respectively) with a significant interaction effect (Wald Chi-Square = 4.31, p = 0.03) (H2 confirmed). For relative-value tasks (T4 and T5), only the normalized mode had a significant effect (Wald Chi-Square = 5.10, p = 0.02) (H3 confirmed).

6.6.2 Completion Time

We compared the completion time (in seconds) for each task using a mixed-model ANOVA with repeated measures. For the retrieve-value task, on average, the completion time (sec) for each interface was for jitter 44.26, absolute 56.84, normalized 52.45, and both 56.57. There was no significant difference between interfaces (p > 0.05).

For the absolute-value task (T2 and T3), on average, the completion time (sec) for each interface was for jitter 30.74, absolute 32.3, normalized 33.6, and both 47.91. The interface had a significant main effect (F(3, 207) = 11.5, p < 0.01).

For relative-value task (T4 and T5), on average, the completion time for each interface was for jitter 26.6, absolute 31.12, normalized 31.38, and both 46.78. The interface had a significant main effect (F(3, 207) = 10.12, p < 0.01). However, when we conducted pairwise comparisons with adjusted p-values using simulation, the only significant difference in time spent was when using the both interface, which took longer (p < 0.01 for all comparisons).

6.6.3 Confidence

The participants self-reported level of confidence was reported using a 7-point Likert-scale rating. For the value-retrieving task (T1), a Kruskal-Wallis non-parametric test revealed that the type of interface had significant impact on the confidence level (χ²(3) = 74.57, p < 0.01). The mean rating for each interface was for jitter 4.8, absolute 6.3, normalized 6.0, and both 6.25. A post-hoc Pairwise Wilcoxon Rank Sum test was employed with Bonferroni correction to adjust for multiple comparisons. The jitter interface was significantly lower than the other three modes (p < 0.01 for all cases). There was no difference between absolute, normalized, and both interfaces.

For absolute-value tasks (T2 and T3), a Kruskal-Wallis non-parametric test revealed that the type of interface had significant impact on the confidence level (χ²(3) = 18.32, p < 0.01). The mean rating for each interface was jitter 5.4, absolute 5.7, normalized 5.0, and both 5.8. A post-hoc Pairwise Wilcoxon Rank Sum test was employed with Bonferroni correction to adjust for multiple comparisons. The interface with both modes was significantly higher than normalized and jitter mode (p < 0.01 for both); however, there was no difference with the absolute mode. The interface with absolute mode was significantly higher than normalized and jitter mode (p < 0.01).

For relative-value tasks (T4 and T5), a Kruskal-Wallis non-parametric test revealed that the type of interface did not have significant impact on the relative tasks (χ²(3) = 4.1, p = 0.2). The mean rating was jitter 4.7, absolute 4.9, relative 4.9, and both 4.8.

One possibility for explaining this result is that the relative task is more difficult than the other tasks. The low correct percentage of questions are also shown in Figure 9. To see that, we tested the confidence level between task types. A Kruskal-Wallis non-parametric test revealed that the type of task had significant impact on the confidence level (χ²(2) = 148.1, p < 0.01). The mean rating for retrieving value 5.9, absolute 5.5, and normalized 4.8. The post-hoc Pairwise Wilcoxon Rank Sum test was employed with Bonferroni correction to adjust for multiple comparisons, and showed that all three task types were significantly different (p < 0.01 for all cases).

7 Discussion

Summarizing our results from the user study confirms that gatherplots outperform jittering for both of the object metrics—accuracy and completion time (albeit only partially for the latter)—as well as for the subjective confidence measure. While the dataset in the study was only moderately sized, there was still sufficient overplotting to make normal scatterplots useless, causing us to use jittering as the baseline.

Overplotting in scatterplots is a well-known problem, and several other efforts address it, such as splatterplots [8] and GPLOMs [9]. Compared to these works, our main contribution is that gatherplots preserve the identity of marks. Based on the clutter reduction framework proposed by Ellis and Dix [5], gatherplots are distortion-based methods, while KDE, histograms, and violin plots are appearance-based. In this section, we will discuss trade-offs compared to appearance-based methods, as well as situations when gatherplots are appropriate and not. We also discuss our user study.
7.1 Scalability

As datasets become larger, the scalability of a visualization becomes an important issue. Scatterplots support two main tasks: detecting correlations as well as outliers. Gatherplots are effective in showing correlations as the dataset grows—as shown in Figure 10(a), (b), and (c)—but this also causes the dot size to shrink, which makes detecting outliers becomes less plausible. Splatterplots [8] handle this by using two different visual representation for dense areas and sparse areas, whereas gatherplots have no such mechanism. In gatherplots, the relative mode enlarges outliers to fill the assigned space. This makes spotting outliers in gatherplot easier (Figure 10(e) and (f)), as well as to compare distributions (Figure 10(d)). As datasets grow in size, individual object identification becomes less relevant, and gatherplots begin to resemble histograms or violin plots.

Even though gatherplots can theoretically be extended to virtually unlimited datasets, the practical bottleneck lies in calculating layouts for individual objects, which requires heavy computation compared to appearance-based methods. Also, the memory required to handle individual object independently is another bottleneck. These computations are hard to justify for a large dataset, because the amount of output information is nevertheless same. In this sense, gatherplots are not scalable to large datasets. However, according to Shneiderman’s visual information seeking mantra—overview first, zoom and filter [39]—even for large datasets, zooming can make the application of gatherplots desirable.

7.2 Named vs. Anonymous Objects

In some multidimensional datasets, data points have names, whereas in others they do not. For example, in datasets containing entities such as cars or digital cameras, it is a common task for users to search cases that suit their needs. For this case, maintaining the identity of individual data points is important because it enables brushing and linking more easily than for aggregated forms such as histograms. There are other datasets, such as the famous iris flower dataset, where individual data points do not have names. For such datasets, the benefit of maintaining object identity can instead be explained using frequency grids. According to Cosmides and Toody [24], the concept of relative percentage is a new concept in human evolution, and explicitly showing distribution using discrete countable objects makes assessing the relative percentage easier and more comfortable.

7.3 Visualizing Normalized Data

According to Im et al. [9], one tradeoff in designing GPLOMs was whether axes of the same variable should be scaled to the same range or not. If scaled to the same range, it would be easy to compare to adjacent charts, but results in large vacant spaces for sparse areas. This happens when there are severe imbalances in data distribution. Both options are valid depending on the task at hand, but it is difficult to represent it visually in histograms so that users can see it.

In gatherplots, these two options are supported as absolute vs. normalized modes. Because we show the individual objects as separate visual marks, it is feasible to deliver this information more explicitly. If the size of all data points are the same, users will understand that they all use the same scale, while rendering points with different sizes conveys the information that they are normalized.

7.4 Limitations of Evaluation Study

Although scatterplots support several types of tasks such as detecting correlation, clusters, or outliers, in our experiment we decided to test a particular case with categorical data, which has distinctive views compared to conventional scatterplots. Even if this is a narrow case, the purpose of our study was to show the effectiveness of different layout modes in a quantitative way. The results indicated that the users could understand the visualization and accomplish the tasks that should be supported. However, we also observed that the difficulty level was different for each task type. In general, ranking tasks were more difficult than comparison tasks, and questions asking about relative values were more difficult than those about absolute values. Therefore, maintaining similar difficulty level among tasks should also be considered while designing future comparative evaluations such as this.

In our study, we selected scatterplots with jittering as the baseline for comparison because (1) it extends scatterplots to manage overplotting, (2) it maintains individual objects, and (3) it is a well-known technique. However, for future studies it would be also desirable to compare the performance with a purpose-specific technique, such as basic scatterplots, histograms, or hieraxes.

8 GatherLens: A Gathering Magic Lens

Scatterplots have a familiar layout and an intuitive continuous scaling in the Cartesian space defined by the graphical axes, whereas gatherplots introduce discontinuities that
may make them more difficult to understand. However, gathering does not necessarily have to be applied globally. Instead, we here propose a local application of gathering in a Magic Lens technique [10] that we call GatherLens (Figure 11).

A Magic Lens is a user-controlled interaction tool that changes the visual representation of the underlying graphical object it overlays [10]. The GatherLens is accordingly a Magic Lens that applies local gathering in a scatterplot to the data points that it overlaps. This gives the user the ability to selectively manage overplotting in specific areas in a scatterplot without changing its overall visual representation.

Lens geometry gives us additional options for layout of the stacked groups in the lens (each is visible in Figure 11):

- **Standard lens**: A rectangular lens that applies standard gathering to the contained points.
- **Histogram lens**: Here, the stacked groups are arranged and aligned so that they resemble a histogram.
- **Pie lens**: Similarly, the group layout here is radial and centered, yielding a pie or donut layout.

### 9 Conclusion and Future Work

We have proposed the concept of the gather transformation, which enables space-filling layout without overdrawing while maintaining object constancy. We then applied this transformation to scatterplots, resulting in gatherplots, a generalization of scatterplots, which enable overview without clutter. While gatherplots are optimal for categorical variables, it can also be used to ameliorate overplotting caused by continuous ordinal variables. We discussed several aspects of gatherplots including layout, coloring, tick format, and matrix formations. We also evaluated the technique with a crowdsourced user study showing that gatherplots are generally more effective than jittering, and absolute and relative mode serve specific types of tasks better. We also applied the gathering transformation to a Magic Lens interaction for local control; this lens has three different layout modes.

We believe that gathering is a general framework that captures the transition between overlapping and space-filling visualizations while maintaining object identities. In the future, we plan on studying the application of this framework to other visual representations. For example, overplotting is a common problem when visualizing categorical variables in a parallel coordinates plot. Parallel sets aggregate elements for the same value of a categorical variable into blocks, but loses the identity of objects. By applying the gathering framework, parallel sets can be reconstructed to render individual lines instead of block
ACKNOWLEDGMENTS

We would like to thank S. Karthik Badam, Jungu Choi and for helpful discussions and Senthil K Chandrasegaran and Ji Soo Yi for evaluation. Also, we would like to acknowledge all the help we received from communicating with a number of visualization creators, some of whose works are cited in the paper.

REFERENCES

Deok Gun Park received a bachelor in electrical engineering in 2000 and a master's degree in biomedical engineering in 2002 at Seoul National University, Seoul, South Korea. Currently, he is pursuing a Ph.D. degree in computer science at University of Maryland, College Park, MD, USA.

Sung-Hee Kim the Ph.D. degree in 2014 from the School of Industrial Engineering at Purdue University, West Lafayette, IN, USA. She received a bachelor and master degree in Computer Science and Engineering at Ewha Womans University, South Korea, in 2006 and 2008.

Niklas Elmqvist received the Ph.D. degree in 2006 from Chalmers University of Technology in Göteborg, Sweden. He is an associate professor in the College of Information Studies at University of Maryland, College Park, MD, USA. He was previously an assistant professor in the School of Electrical & Computer Engineering at Purdue University in West Lafayette, IN. He is a senior member of the IEEE and the IEEE Computer Society.