Towards Fluent Interactive Data Visualization

Adam Bartonicek

2023-08-10

Contents

0.1 Introduction

There is a subtle yet profound issue in the production and use of interactive data visualizations: *when we interact with a plot, what exactly are we interacting with?* On the face of it, it may not even seem obvious that there is any problem. Like static plots, interactive plots consist of geometric objects such as points, lines, or areas. Surely, when interacting with a visualization, we can just interact with the geometric objects we see? However, this is rarely the case. The reason is that the geometric objects we see cannot be interpreted in isolation, but only with reference to some underlying data. Further, when plotting data, we rarely plot it directly. Instead, most plots use the geometric objects to display some statistical summaries of the data, such as count, sum, mean, or quantiles. These summaries are obtained by applying mathematical functions to the data. The core argument of the present text is that these mathematical functions (and their properties) impose limits on what kinds of visualizations and interactions can be meaningfully composed. Before diving deeper, however, let's first define some key terms and draw a rough sketch of the data visualization process.

To create a data visualization, be it static or interactive, we need several key ingredients: data, summaries, scales/coordinate systems, and geometric objects. Firstly, every data visualization is built on top of some underlying data. Typically, the data is stored in a tabular(or "tidy" [CITE, trigger Simon]) format which we can here represent as a set of rows R (this is not strictly necessary, but will be used here for convenience). Secondly, each row $r_i \in R$ is transformed via a function h into a collection of summaries $s_j \in S$. The function may be one-to-one (bijection), as in the case of a scatterplot, or, more often, many-to-one (surjection), as in the case of plots such as barplot, histogram, density plot, or violin plot. The function will typically reduce the cardinality of the set at hand such that $|S| \leq |R|$ (i.e. in a typical barplot, there will be fewer bars than there are rows of the data). This is done by stratifying on one or more variables which may either come from the data directly (as in the case of a barplot or a treemap) or may themselves be a summary of the data (as in the case of histogram bins). Importantly also, each collection $s_j \in S$ may (and usually will) hold multiple values produced by a different constituent function each - for example, the collection s_j of summaries for a single boxplot "box" will consist of a median, first and third quartile, and the minimum and maximum of some variable, for a given level of some stratifying variable (which itself is an element of s_j). The output of these constituent functions may also depend on some external parameters, such as anchor and binwidth in a histogram. Combined, the summaries form a partition of the data - if we give each collection of summaries a label $j \in J$ such that $|J| = |S|$, we could split the data rows $r_i \in R$ into sets belonging to the same collection of summaries R_j (with no overlapping parts: if $j \neq k, R_j \cap R_k = \varnothing$), such that, if we combine these sets, we recover the original dataset: $R = \bigcup_{j \in J} R_j$. Thirdly, each collection of summaries $s_j \in S$ needs to be translated from the data- (or summary-) coordinates to graphical coordinates/attributes $g_j \in G$, by

transforming each value via a function i typically called a scale (note that this mapping preserves cardinality: $|G| = |S|$). For numeric summaries, scales typically come in the form of linear transformations, such that the minimum and maximum of the data are mapped near the minimum and maximum of the plotting region, respectively. Scales may also provide additional non-linear transformations such as log-transformation or binning. Finally, the collections of graphical coordinates $g_j \in G$ are displayed by drawing them as geometric objects inside the plotting area. The geometric objects may be simple, such as points, lines, or bars, or compound, such as a boxplot or pointrange. Importantly, each attribute necessary to draw a geometric object such as x- and y-position, width, height, area, etc... needs to be present in the corresponding g_j.

The whole process can be summarized as such:

$$R \overset{h}{\to} S \overset{i}{\to} G$$

Or, equivalently:

$$\text{(rows)} \overset{\text{summarize}}{\longrightarrow} \text{(summaries)} \overset{\text{translate/encode}}{\longrightarrow} \text{(graph. coordinates)}$$

The above should be fairly non-controversial description of how a data visualization is produced, and applies equally well to static as well as interactive visualizations. However, it should be mentioned, in some treatments of data visualization, the summarizing step (i.e. $h : R \to S$) is often deemphasized, in one of two ways. In the first case, the data is already presumed to arrive pre-summarized, i.e. $S = R$, and data visualization is just about encoding the data into graphical coordinates. This framing feels especially natural when considering plots which show a 1-to-1 mapping (bijection) between the data and the geometric objects, such as the scatterplot or lineplot/time-series chart. Indeed, in these plots, the summarizing function is identity, and as such *could* be ignored. In the second case, the computation of summaries is considered, however, it is absorbed into the translation step, such that $h = i$ (e.g., a histogram may be thought of as directly translating its underlying variable into bin coordinates). Both of these approaches produce a tight coupling between the summaries and graphical coordinates.

However, we can give a much general account of visualizations by treating the summarizing and translating steps separately. Some plots may translate the same summaries into very different sets of graphical coordinates and encode them through different objects; others may compute different summaries but display them the same way. As an example of the former, the histogram and spineplot both rely on the summary of binned counts of some underlying continuous variable x, however, histogram encodes the bin breaks into the bin boundaries along the x-axis and the counts as the top bin boundary along the y-axis, whereas spineplot encodes the counts into both x- and y-axis bin boundaries, with the x- dimensions stacked and y-dimension scaled to 1.

There are also a couple of important caveats. Firstly, while not always the case, *order* can be important. Within each $s_j \in S$ and $g_j \in G$, some summaries may be ordered lists of values, as in the case of the lineplot (i.e. lines need to be drawn by connecting a series of points in the correct order). Secondly, the collections in S (and in turn, in G) may themselves be ordered and form a hierarchical or tree-like structure. This is particularly relevant in the case of stacking. For example, when drawing a stacked barplot, we need to stack the graphical coordinate values representing the sub-bars on top of each other, such that each sub-bar is stacked under the appropriate parent-bar and in the correct order (e.g. such that sub-bars belonging to group 2 are always stacked on top of those belonging to group 1). Finally, the data limits (minimum and maximum) and values that are used for scales can often be derived from S rather than from the raw data (e.g. the upper y-axis limit in barplot or histogram is the highest count across bins). Further, the limits may come from a higher level of hierarchy than the summaries that are actually being drawn - for example, in a stacked barplot, for the upper y-axis limit we need to know the count (height) of the tallest *whole* bar but do not need to know the counts within the sub-bars (since these are, by definition, smaller).

This is where the differences between static and interactive visualizations become noticeable. In static visualizations, all computation is done only once, before the plot is rendered, and so the issues of order, hierarchical structure of the summaries, and the tracking of axis limits are less important. Interactive

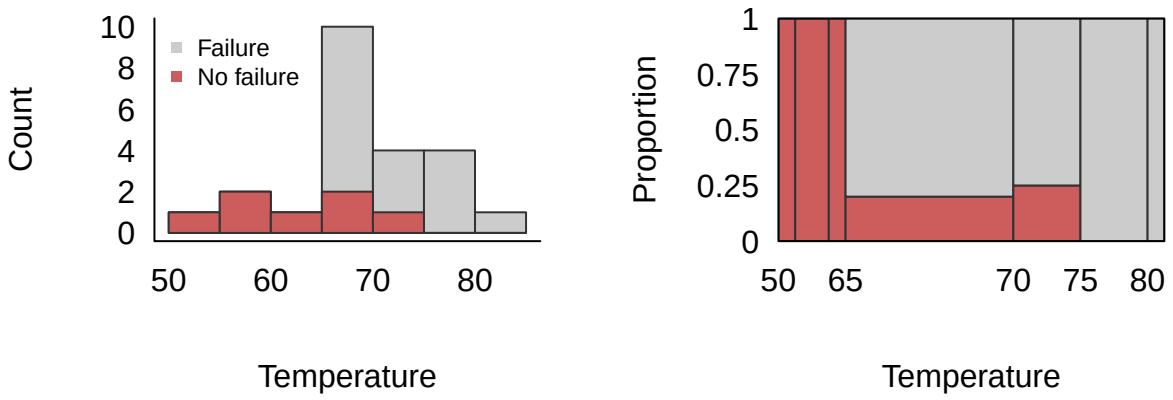

Figure 1: Histogram and spineplot use the same summaries but encode them in different ways. In histogram (left), the x-axis display bin breaks and y-axis displays count, stacked across groups (failure/no failure). In spineplot (right), the x-axis and y-axis both display count; the x-axis shows count stacked across bins, whereas the y-axis shows count stacked across groups and scaled by the total bin count. The bin breaks are displayed as x-axis labels (however, the underlying summary is still stacked count).

visualizations, on the other hand, need to reactively respond to the user's input, and some computations may need to be run many times within a single second. As a result, it is imperative to organize the process in such a way that only as little work as is neccessary is done.

0.2 Interactivity and Hierarchy

0.3 The Problem of Statistical Summaries

There is an even deeper issue when it comes to interactivity and statistical summaries of the data. Specifically, not every statistical summary "works" equally well - instead, some summaries may be better than other. Let's first illustrate the problem with an example.

Linked brushing or highlighting is one of the most popular types of interactive features used in interactive data visualizations [CITE]. It allows the user to select objects (such as points or bars) within one plot by e.g. clicking or clicking-and-dragging, and the corresponding cases (rows of the data) are then highlighted in all other plots. Its usefulness comes from the fact that allows the user to rapidly "drill-down" [CITE] and explore the summaries that would result from subsetting different rows of the data, within the context of the entire dataset.

Now, let's imagine we have three interactive plots: a classical scatterplot, a barplot of summarizing the count of cases within the levels of some categorical variable x, and a barplot that summarizing the mean of some other variable y, within levels of the same categorical variable x. The plots are linked such that they allow for linked brushing/highlighting. Intuitively, it might seem that the barplot of counts and the barplot of means are equally valid/useful representations of the underlying data. However, if we consider these plots in the context of linked brushing, few subtle-yet-fundamental questions become apparent.

Firstly, as is shown in Figure 2, how do we draw an empty selection? In the case of counts, we have a meaningful default value - zero - as in "the number of cases in an empty set is 0". However, there is no similar default value for means: the mean of an empty set is not defined. We could just not draw

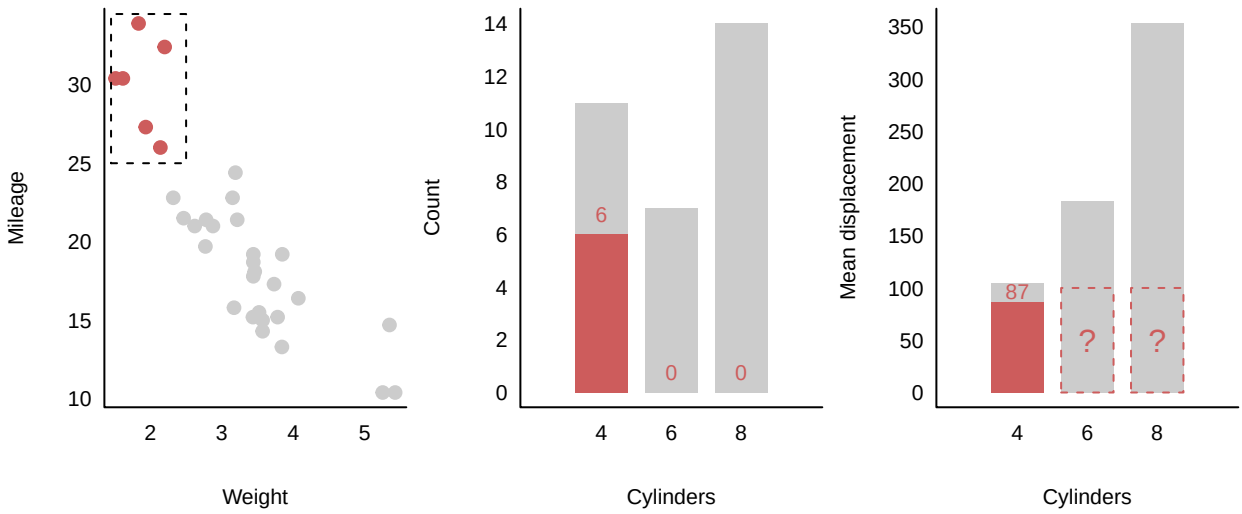

Figure 2: In the barplot of counts (left), the count within a selection (red) is zero if the selection is empty and so an absence of a bar indicates a count of zero. In the barplot of means (left), the mean of an empty selection is not defined and so we do not have a clear way to represent empty selection. Absence of a bar could indicate either no cases are selected or some cases are selected and their mean is equal to the lower y-axis limit.

the bar representing the empty selection, however, that decouples the statistical summary from the visual representation: the absence of a bar may now indicate that *either* no cases are selected *or* that some cases are selected and their mean is equal to the lower y-axis limit.

Second, as shown in Figure 3, how does the summary on the selection relate to the summary on the whole? In the case of the barplot of counts, the height of the sub-bar is always less than or equal ($\leq$) to the height of the whole bar (because so is the count). In this way, we can always draw the sub-bar on top of the whole bar, and the whole bar act as a stable visual reference - the outline of the whole bar will stay the same no matter which cases of the data are selected. In fact, we can either draw the whole bar (as shown in grey in Figure 3) and draw the selected sub-bar (red) over it, both starting from the y-axis origin (0), or we can draw a red sub-bar and the "leftover" grey-sub bar stacked on top of it, starting from the top y-coordinate of the selection bar and with height = count of the whole − count of the selection. The resulting plots are visually identical. However, the barplot of means does not share these nice properties. Importantly, the "sub-bars" can be taller than the whole bar (because the mean of a subset can be greater than the mean of the original set). As a result, if we draw the selection sub-bars on top of the whole bar, the whole bar may be completely obscured by it (as shown in Figure 3). We could choose to draw the selection sub-bars as semitransparent, or draw the the bars side-by-side instead of on top of each other, however, the question then remains how to display the non-selected (grey) cases - do we draw the "whole" bar that remains the same height throughout interaction, or the "leftover" bar that changes as the selection changes? Also, if we draw the bars side-by-side, will the whole bar be initially wide and shrink in response to selection to accommodate the selection bar, or will it be narrow from the initial render?

Finally, as displayed in Figure 4, when multiple selections/groups are present, how do we combine them together? Again, in the case of the barplot of counts, there is an idiosyncratic way to do this: we can stack the counts across the selection groups and the corresponding bars on top of each other, and the height of the resulting bar will accurately represent the count of the whole. And, as in the case of a single selection group, we can either draw the bars over each other, starting from the y-axis origin, or draw sub-bars stacked on top of each other, each starting where the last left off, and the resulting plots will be identical. In the case of barplot of means, however, there is no meaningful way to combine the selections. If we were to stack the

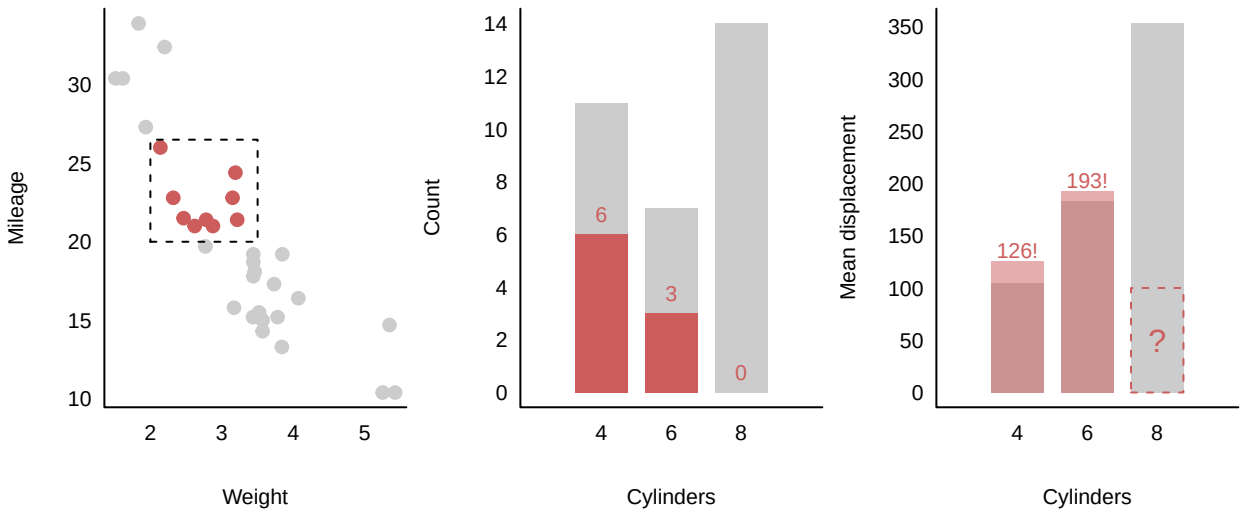

Figure 3: In the barplot of counts (middle), the count within a selection (red) is always less than or equal to the count within the whole and the outline of the bars does not change. In the barplot of means (left), the mean of a selection can be greater and so the outline of the bars can change.

bars on top of each other, the resulting statistic (i.e. the sum of the group means) would not be meaningful. Worse yet, if we were to take the mean of the group means, this might (and most likely will) be different from the mean of the whole: the mean of the group means is different from the grand mean. And again, the mean of the group means may be less that any one of the group means. Since we cannot meaningfully combine the statistics, we could draw them side-by-side, but again, considerations about rendering the base group and the bar width present themselves.

To summarize, counts provide:

- A meaningful and unambiguous way to display empty selections (0 or absence of a bar)
- A stable visual reference (the count within the whole bar is always greater than or equal to the count within the sub-bars)
- A meaningful way to combine the statistics together (the sum of the counts within sub-bars is always equal to the count within the whole bar).

Means have none of these nice properties. There are of course are ways to display means with linked brushing/selection, with some having been described above. The key point, however, is that with means there are more considerations and decisions that need to be made to produce coherent interactive visualizations. Consequently, a person interacting with a linked figure containing a barplot of means may be surprised in how the plot behaves, since among the many answers to each of the three questions there are none which would be more "natural" than others. In the barplot of counts, however, there *are* such natural answers to the questions.

Symmetric monoidal preorders also come with some4 computational advantages. Firstly, these summaries are guaranteed to be computable within a single pass through the data, and can also be updated online if new data arrives, without having to refer to the old. This does not necessarily make them superior to other summaries, such as mean or variance, which can also often be computed by collecting multiple constituent summaries in a single pass through the data and them combining them together in one step after (such as sum and count for the mean), and for which online algorithms also often exist too. However, the advantage is that every symmetric monoidal preorder is automatically single pass computable and online - we will never need to look for an algorithm. Secondly, if our plot implements reactive axis limits, since the count within

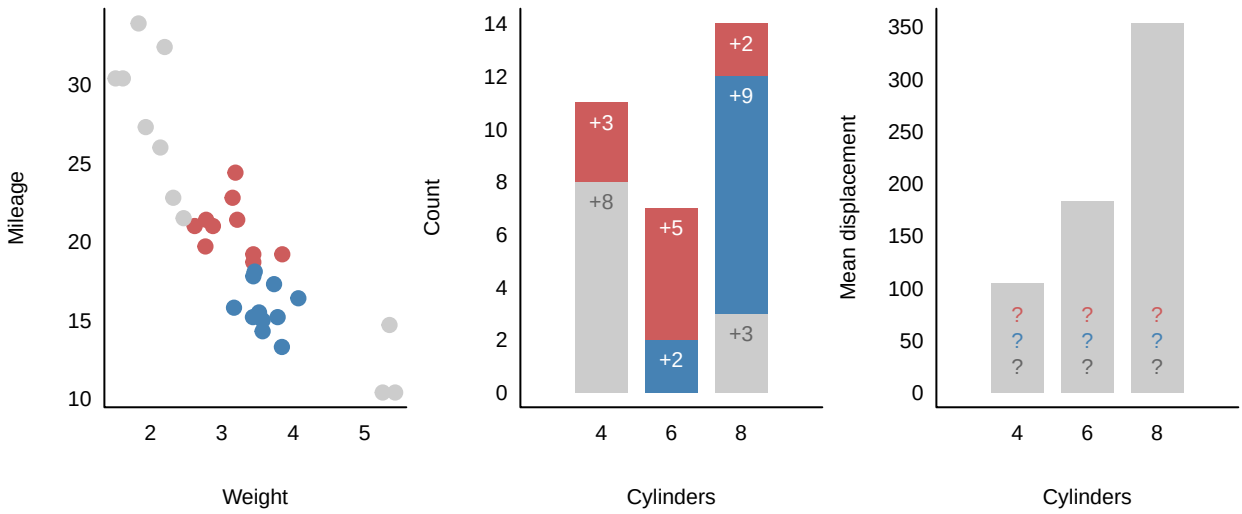

Figure 4: In the barplot of counts (middle), statistics can be stacked on top of each other such that the count of the stacked bar accurately represents the count of the whole bar (i.e. one that would result from no selections). In the barplot of means, no such procedure for combining statistics exists.

the selection is always guaranteed to be less that or equal to the count within the whole, we only need to keep track of the counts on the whole. For example, in the case of the barplot of counts, we know that the upper y-axis limit will always be equal to the height of the tallest whole bar, and we can safely ignore the sub-bars. If there are N levels of the categorical variable and K selection groups, there will be N whole bars and $N \times K$ sub-bars. This difference might unimportant if N and K are small (as they are likely to be, in the case of the average barplot). However, if there are, for example, two categorical variables with N, and M levels, respectively, as in the case of a fluctuation diagram/bubbleplot/treemap, and both N and M have many levels, then having to iterate through all $N \times M \times K$ sub-bar summaries every time an interaction happens might become prohibitive.

One might wonder whether these questions arise only in the case of the linked brushing.