Towards Fluent Interactive Data Visualization

Adam Bartonicek

2023-10-30

Contents

1 Introduction

Interactive data visualization (IDV) has seen a rapid growth in popularity over the last few decades. From the humble roots of early, highly specialized systems, such as those of Fowlkes (1969) and Kruskal (1965), there has been a steady progress towards more and more general and feature-rich frameworks. The first really "general-purpose" system was *PRIM-9* (Fisherkeller, Friedman, and Tukey 1974), which allowed for exploration of high-dimensional data in scatterplots using projection, rotation, subsetting and masking. Later systems, such as *MacSpin* (Donoho, Donoho, and Gasko 1988), *Lisp-Stat* and *XLisp-Stat* (Tierney 1989, 2004), and *XGobi* (Swayne, Cook, and Buja 1998) provided rich features such as interactive scaling, rotation, linked selection (or "brushing"), and interactive plotting of smooth fits in scatterplots, as well as interactive parallel coordinate plots and grand tours. They were later followed by other systems such as *Mondrian* (Theus 2002), *GGobi* (Swayne et al. 2003), *iPlots* (Urbanek and Theus 2003), and *cranvas* (Xie, Hofmann, and Cheng 2014). Alongside these later developments, which have largely come from the field of statistics, the rise of Web technologies and interactive web apps has spawned its own family of IDV systems. Among these, the earlier systems such as *Prefuse* (Heer, Card, and Landay 2005) and *Flare* (Burleson 2020) relied on external plugins (Java and Adobe Flash Player, respectively) while later systems became truly Web-native by embracing JavaScript. Among them, *D3*.js (Bostock, Ogievetsky, and Heer 2011) has grown to great popularity, alongside its high-level interface *plotly*.js, (Plotly Inc. 2022), and so have other frameworks such as *Vega* (Satyanarayan et al. 2015), *Vega-lite* (Satyanarayan et al. 2016), *Altair* (VanderPlas et al. 2018), and *Highcharts* (Highcharts Team 2023).

Interactive figures now frequently appear in online news articles, business dashboards, and scientific publications and blogs. Yet, despite the proliferation of IDV systems and various taxonomies of interactive features (see e.g. Yi et al. 2007), there seems to remain a lack of a general framework for reasoning about interactive data visualizations, in the way that e.g. *grammar of graphics* (Wilkinson 2012) provided a way to reason about static visualizations. That is, interactive data visualizations, like all visualizations, require us to process data into statistical summaries that can then be translated into screen coordinates and drawn. However, different types of interaction require vastly different . Some interactions can be implemented by manipulating graphical attributes of the visualization only. For example, to implement panning or zooming,

all we need to do is compute the axis limits from the data once. After that, we can forget the original data and merely update the four scalar values (= lower and upper x-axis and y-axis limits) whenever the user performs the requisite actions. However, this is not the case for other, more complex types of interactions. For example, if we want to implement an interactive histogram in which the user can change the binwidth and anchor, we need a way to recompute the number of cases in each bin after either of the two parameters is updated. This means that we have to refer to the original data. Similarly, to implement linked highlighting/brushing, whereby the user can highlight some cases in the data across multiple plots by either clicking or click-and-drag-selecting the corresponding geometric objects in one plot, we need to keep track of which cases belong to which object.

How should we structure the process of computing summary statistics and translating them into graphical coordinates, such that we can do this efficiently in response to user input? Wickham et al. (2009) have called for an IDV pipeline or "plumbing". This call has, despite some attempts (e.g. Xie, Hofmann, and Cheng 2014), been left largely unaddressed.

2 The illusion of objects

The key towards a general IDV framework may lie in a subtle yet profound question that inevitably appears in the production and use of interactive data visualizations: *when we interact with a plot, what exactly are we interacting with?* On its face, it may seem trivial. A person clicking a bar in an interactive barplot may be convinced that they are interacting with the coloured rectangle on the screen, since, by design, that is the salient "thing" they see change in front of them. And in some way, this is true - by interacting with the bar, we can affect its graphical attributes: we can change its colour, we can squeeze it/stretch it, and so on. Yet, in another, deeper way, this perception of interacting with a plain geometric object is just an illusion. How so?

The illusion lies in the fact that the bar is not just the geometric object - the coloured rectangle - it is represented by. Instead, the rectangle is only ever meaningful as a "bar" within the context of the plot. We can see this quite easily - if we were to take the coloured rectangle outside of the plot, we would lose some crucial information that the rest of the plot provides.

Figure 1: A rectangle is not a bar in a similar way that a painting of a pipe is not a pipe.

Thus, objects in a plot are imbued with some additional information or structure, beyond their simple geometry. That statement should not seem surprising or controversial to people familiar with data visualization, and indeed geometric objects are considered just one "layer" in, e.g. the *grammar of graphics* (Wilkinson 2012). However, it may be more challenging to define in detail what exactly this "structure" is. There are a few ideas we may be able to muster. First of all, we know that the geometric objects in plots are supposed to represent some underlying data. That much is clear - if the objects in a graphic do not represent any external data but are instead drawn according to some arbitrary rules, we cannot really, in good conscience, call the resulting graphic a "plot". But data is only a part of the story.

When drawing plots, we rarely represent the raw data directly. Instead, we often summarize, aggregate, or transform. We do this by applying mathematical functions such as count, sum, mean, log, or the quantile function. And it is the output of these transformations that we then represent by the geometric objects.

So, when interacting with a bar in an interactive barplot, we do not just interact with a plain geometric object. Instead, we interact with a mathematical function, or, in fact, several of them. This is very important since mathematical functions have properties, and these properties impose limits on what kinds of visualizations and interactions we can meaningfully compose. This is the crux of the argument presented in this text. Before diving deeper, however, let's first define some key terms and draw a rough sketch of the data visualization process as a whole.

3 Rough sketch of the data visualization process

To create a data visualization, be it static or interactive, we need several ingredients: data, summaries, scales/coordinate systems, and geometric objects. These should be familiar to most users of interactive data visualization systems. However, it may still be useful to lay them out in order, and examine the specific features and quirks of each one of them.

First of all, as was mentioned in the previous section, every data visualization needs to be built on top of some underlying data. We can represent this as a set of some arbitrary units of information (data) D. Data in the wild usually comes with more structure than that - for example, we often encounter data stored in a tabular (or "tidy" (Wickham 2014)) format, stratified by rows and columns. In that case, we could substitute D by the set of rows R, the set of columns C, or the set of cell values $R \times C$ (where $\times$ indicates the cartesian product). However, for the purpose of this broad description, we do not have to assume any special structure and just speak of the data units $d \in D$.

Secondly, at some point during the visualization process, we need to transform the set of data units D into a set of collections of summaries S via a function α. The summarizing function α can have many different flavours. It may be the case that α is one-to-one (bijection), in which case there is one summary for every unit of data (and vice versa). This is the case, for example, with the prototypical scatterplot, in which α is just the identity function (every unit of data/row gets assigned one "point"). However, more often, α is many-to-one (surjection), which means that each summary may be composed of multiple units of data. Examples of this include the typical barplot, histogram, density plot, or violin plot. When α is many-to-one, it will typically reduce the cardinality of the data, such that $|S| \leq |D|$ (e.g. in a typical barplot, there will be fewer bars than there are rows of the data, unless each row represents a unique level of the categorical variable). To turn n units of data into k collections of summaries, we need to somehow stratify the data on one or more variables. These variables may either come from the data directly (i.e. the variables used are "factors", 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 of summaries $s \in S$ may (and usually will) hold multiple values, produced by a different constituent function each - for example, the collection s for a single boxplot "box" will consist of the median, the first and third quartile, the minimum and maximum, and the outlier values of some variable, all for a given level of some stratifying variable (which itself will also be an element of s). Finally, the output of these constituent functions may also depend on some external parameters, which may be either directly supplied by the user or heuristically inferred from the data by the visualization system itself. Examples of such external parameters include anchor and binwidth in a histogram.

Thirdly, each collection of summaries $s \in S$ needs to be translated from the data- (or summary-) coordinates to graphical coordinates/attributes $g \in G$, via a function β. This means that each summary value gets mapped or "scaled" to a graphical attribute via a constituent scaling function. Note that this mapping preserves cardinality - there are as many collections of graphical attributes as there are collections of summaries, $|G| = |S|$. For numeric/continuous summaries, scales often 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. Continuous scales may also provide non-linear transformations such as the log-transformation or binning, and the values may even get mapped to discrete graphical attributes

(e.g. binned values may get translated to one of 5 different shades of a colour). Likewise, discrete summaries can be translated to either continuous (e.g. position) or discrete (e.g. colour) graphical attributes. There are also coordinate systems, which may further translate values from multiple scales simultaneously, e.g. by taking values in cartesian (rectangular plane) coordinates and translating them to polar (radial) coordinates. Either way, β can be viewed as one function, representing the composition of any number of scales and coordinate systems applied to various summaries.

Finally, the collections of graphical attributes/coordinates $g \in G$ are drawn as geometric objects inside the plotting region, which we can represent as the set of pixels P. While the act of drawing does take the collections of graphical coordinates $g \in G$ as inputs, it does not simply return an output for each input (like a mathematical function would), but instead mutates the state of the graphical device via a side effect γ^*, i.e. changing the colour values of pixels in P. In other words, how the graphic ends up looking may depend, for example, on the order in which we draw the objects. For example, when drawing points of different colour, some points may end up being plotted over others, and thus the final result may depend on whether we draw red points first and yellow second or vice versa. As such, γ^* is not a simple mapping from G to P and we cannot call it a true mathematical function (since that would require it to merely assign an output to each input). 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 the geometric object, such as x- and y-position, width, height, area, etc... needs to be present in the corresponding g.

The whole process can be summarized as follows:

$$D \overset{\alpha}{\to} S \overset{\beta}{\to} G \overset{\gamma^*}{\Rightarrow} P(\tag{1}$$

Or, equivalently:

$$\text{(data)} \overset{\text{summarize}}{\longrightarrow} \text{(summaries)} \overset{\text{translate/encode}}{\longrightarrow} \text{(graph. coordinates)} \overset{\text{draw}^*}{\Rightarrow} \text{(graph. device state)} \tag{2}$$

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.

References

Bostock, Michael, Vadim Ogievetsky, and Jeffrey Heer. 2011. "D^3 Data-Driven Documents." *IEEE Transactions on Visualization and Computer Graphics* 17 (12): 2301–9.

Burleson, Thomas. 2020. "Flare | Data Visualization for the Web." *Blokt - Privacy, Tech, Bitcoin, Blockchain & Cryptocurrency*. https://blokt.com/tool/prefuse-flare.

Donoho, Andrew W, David L Donoho, and Miriam Gasko. 1988. "MacSpin: Dynamic Graphics on a Desktop Computer." *IEEE Computer Graphics and Applications* 8 (4): 51–58.

Fisherkeller, Mary Anne, Jerome H Friedman, and John W Tukey. 1974. "An Interactive Multidimensional Data Display and Analysis System." SLAC National Accelerator Lab., Menlo Park, CA (United States).

Fowlkes, EB. 1969. "User's Manual for a System Fo Active Probability Plotting on Graphic-2." *Tech-Nical Memorandum, AT&T Bell Labs, Murray Hill, NJ.*

Heer, Jeffrey, Stuart K. Card, and James A. Landay. 2005. "prefuse: a toolkit for interactive information visualization." In *CHI '05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 421–30. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1054972.1055031.

Highcharts Team. 2023. "Interactive javascript charts library." https://www.highcharts.com.

Kruskal, J. B. 1965. "Multidimensional Scaling." https://community.amstat.org/jointscsg-section/media/videos.

Plotly Inc. 2022. "Part 4. Interactive Graphing and Crossfiltering | Dash for Python Documentation | Plotly." https://dash.plotly.com/interactive-graphing.

Satyanarayan, Arvind, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. "Vega-Lite: A Grammar of Interactive Graphics." *IEEE Transactions on Visualization and Computer Graphics* 23 (1): 341–50.

Satyanarayan, Arvind, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. 2015. "Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization." *IEEE Transactions on Visualization and Computer Graphics* 22 (1): 659–68.

Swayne, Deborah F., Dianne Cook, and Andreas Buja. 1998. "XGobi: Interactive Dynamic Data Visualization in the X Window System." *J. Comput. Graph. Stat.* 7 (1): 113–30. https://doi.org/10.1080/10618600.1998.10474764.

Swayne, Deborah F., Duncan Temple Lang, Andreas Buja, and Dianne Cook. 2003. "GGobi: evolving from XGobi into an extensible framework for interactive data visualization." *Comput. Statist. Data Anal.* 43 (4): 423–44. https://doi.org/10.1016/S0167-9473(02)00286-4.

Theus, Martin. 2002. "Interactive Data Visualization using Mondrian." *J. Stat. Soft.* 7 (November): 1–9. https://doi.org/10.18637/jss.v007.i11.

Tierney, Luke. 1989. "Xlisp-Stat." School of Statistics Report.

———. 2004. "Some Notes on the Past and Future of Lisp-Stat." *Journal of Statistical Software* 13: 1–15.

Urbanek, Simon, and Martin Theus. 2003. "iPlots: High Interaction Graphics for r." In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*. Citeseer.

VanderPlas, Jacob, Brian Granger, Jeffrey Heer, Dominik Moritz, Kanit Wongsuphasawat, Arvind Satyanarayan, Eitan Lees, Ilia Timofeev, Ben Welsh, and Scott Sievert. 2018. "Altair: Interactive Statistical Visualizations for Python." *Journal of Open Source Software* 3 (32): 1057.

Wickham, Hadley. 2014. "Tidy Data." *J. Stat. Soft.* 59 (September): 1–23. https://doi.org/10.18637/jss.v059.i10.

Wickham, Hadley, Michael Lawrence, Dianne Cook, Andreas Buja, Heike Hofmann, and Deborah F Swayne. 2009. "The Plumbing of Interactive Graphics." *Computational Statistics* 24: 207–15.

Wilkinson, Leland. 2012. *The Grammar of Graphics*. Springer.

Xie, Yihui, Heike Hofmann, and Xiaoyue Cheng. 2014. "Reactive Programming for Interactive Graphics." *Statistical Science*, 201–13.

Yi, Ji Soo, Youn ah Kang, John Stasko, and Julie A Jacko. 2007. "Toward a Deeper Understanding of the Role of Interaction in Information Visualization." *IEEE Transactions on Visualization and Computer Graphics* 13 (6): 1224–31.