No More, No Less Than Sum of its Parts: Monoids and The Algebra of Graphics, Statistics, and Interaction

Adam Bartonicek
Department of Statistics, The University of Auckland
and
Simon Urbanek
Department of Statistics, The University of Auckland
and
Paul Murrell
Department of Statistics, The University of Auckland

October 23, 2024

Abstract

Interactive data visualization has become a staple of modern data presentation. Yet, despite its growing popularity, there still remains a lack of a formal framework for turning raw data into summary statistics that can then be displayed by interactive graphics. This gap may stem from a subtle yet profound issue: while we would often like treat statistics, graphics, and interaction as independent entities, they are in fact deeply connected. This paper aims to shed light on this interdependence by relating it to two fundamental concepts from category theory: groups and monoids. Specifically, if we want our graphics to support interactive features which split our data into parts and then combine these parts back together (such as linked selection), then the statistics in our plots need to posses certain algebraic properties. Ultimately, by grounding our thinking in these concepts, we may be able to build more flexible and expressive interactive data visualization systems.

Keywords: Interactive data visualization, category theory, abstract algebra, the Gestalt Principles, descriptive statistics

1 Introduction

> "Effective graphical analysis makes things seem obvious, the effort involved in making the graphical analysis effective is not so obvious." (Unwin, 2018)

The essence of data visualization is comparison. When we visualize, we split our data into parts, summarize each part by one or more summary statistics, encode the summaries into visual attributes such as position, size, or color, and, finally, use these attributes to draw geometric objects (Bertin, 1983; Wilkinson, 2012; for a recent review, see Franconeri et al., 2021; Wilke, 2019). For example, when drawing a typical barplot, we split our data into parts based on the levels of some categorical variable, count the number of cases in each part, and finally represent these counts by drawing bars of corresponding heights. In this way, each geometric object (bar) becomes a representation of its own small data set, a subset of the original data.

To ground this idea in a concrete example that will be used throughout this paper, imagine going through your bookshelf, sorting the books by genre, and stacking them on top of each other. You will construct a physical version of the barplot, forming one "bar" (stack) per genre: one bar for sci-fi novels, one bar for romance, one bar for detective fiction, and so on. Even this very simple example reveals some interesting mathematical properties. For example, a single book cannot go into two stacks (the subsets of the data are *disjoint*) and if all of the stacks are combined back together, we should end up with the bookshelf we started with (the subsets are *surjective* and cover the entirety of the data set, see Ziemkiewicz and Kosara, 2009).

Now, just as we can learn about the contents of our library by sorting books into stacks, we can learn about our data by splitting it into parts. A natural question then arises: can we learn more by subdividing our data into even smaller parts? For instance, in our book

example, we can split each stack of books into smaller stacks based on the gender of the author. Each smaller stack then represents the unique combination (Cartesian product) of the two categorical variables (genre and gender). Moreover, there arises a hierarchical relationship between the stacks: by combining genre-gender stacks (child components), we can recover the original genre stack (parent component). Similar situation often arises in interactive graphics, where we use techniques such as linked selection to highlight specific subsets of our data. Either way, by forming these finer partitions of our data, we can represent it with greater granularity and discover trends that may otherwise hide in the aggregate.

However, once we have subdivided each category into its child components, we need to represent the information in a way that visually preserves this hierarchical relationship. In data visualization, there are two popular methods for doing this, known as "stacking" and "dodging" (see Figure 1). These methods have existed at least since the time of William Playfair (1801; 1822). Let's briefly illustrate them on the example of the barplot. In a stacked barplot, bar segments representing the product of the two factor variables are plotted vertically on top of each other ("stacked"; the operation can be also thought of as "highlighting" parts of the bar). In contrast, in a dodged barplot, bars are plotted side-by-side, as "clusters". Finally, there is also a third more recent method, known as layering, where we plot bar segments in separate graphical layers and use partial transparency to mitigate overplotting.

Much has been written about the relative merits of stacking, dodging, and layering. For example, layering is only useful with few categories, as blending many colors can make it difficult to tell the categories apart (Franconeri et al., 2021; Wilke, 2019). Further, in a landmark study, Cleveland and McGill (1984) showed that people tend to be less

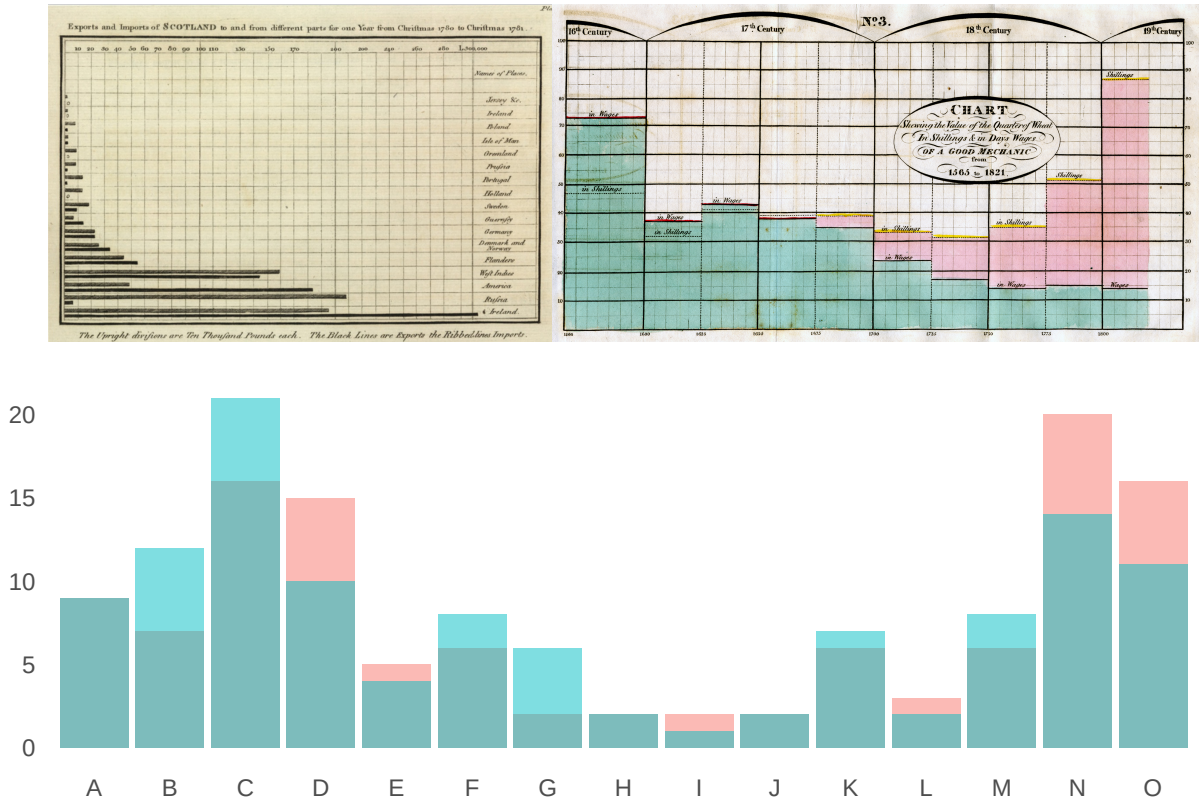


Figure 1: Examples of three methods for dealing with bars that have split into child components (segments). Top row: dodging (left) and stacking/highlighting (right), both by William Playfair (1801; 1822). Bottom row: layering, created with `ggplot2` (Wickham, 2010; the source of the underlying data is the U.S. fuel economy data set, U.S. Department of Energy, 2024).

accurate when reading information from stacked bar charts as opposed to dodged bar charts. Specifically, since the lower y-axis coordinate of a stacked segment is pushed up by the cumulative height of the segments below, it becomes difficult to accurately compare segments' length, both within and across bars (Cleveland and McGill, 1984). Subsequent research has independently validated these findings and expanded upon them (see e.g. Heer and Bostock, 2010; Thudt et al., 2016; Quadri and Rosen, 2021). Due to this suboptimal statistical legibility, many data visualization researchers have urged caution about stacking (see e.g. Byron and Wattenberg, 2008; Cairo, 2014; Franconeri et al., 2021), and some have even discouraged its use altogether (Kosara, 2016; Wilke, 2019).

However, despite the mixed reception towards stacking in static visualization, it remains a popular choice in interactive data visualization. Among popular interactive data visualization libraries, stacked plots such as bar charts and histograms make frequent appearance (see e.g. Forbes et al., 2023; Murray, 2017; The Vega Project, 2022; Vanderplas et al., 2018; Sievert, 2020; Swayne et al., 2003; Xie et al., 2014; Theus, 2002; Urbanek and Theus, 2003; Urbanek, 2011; Wills, 2008; Ware, 2019). What is the reason for this? Has the interactive data visualization community simply failed to catch up with the current best practices? Or is there perhaps more going on?

We argue that not only is stacking a useful technique, despite its limitations, but that it can also offer valuable insights into the mathematical structure underlying our visualizations. To make the strongest case possible, we bring forward evidence from several levels of the visualization process. We start by discussing the properties of visual perception and how they apply to both static and interactive visualizations. Next, we explore the process of computing statistical summaries that form the basis of our plots. Finally, we dive into the algebraic properties of the functions underlying these statistical summaries.

Throughout the text, we use the example of the humble barplot to ground our discussion.

Our goal is to convince you that, in order to produce coherent and well-behaved interactive visualizations, we simply cannot treat graphics, statistics, and interaction as independent components. Instead we need to consider them holistically, within the context of their respective visual and mathematical properties. Understanding this deep web of interdependence may in turn empower us to innovate and build new kinds of robust and expressive interactive visualizations.

2 Visual Perception and Interactivity

2.1 Visual perception and the Gestalt Principles

Over the course of our evolutionary history, the human brain has evolved a sophisticated network for processing visual input. This visual processing pipeline has been the subject of a great body of research in cognitive neuroscience (Goebel et al., 2004; Knudsen, 2020; for a brief overview, see Ware, 2019, 20), and many of its findings are directly applicable to data visualization. For example, certain particularly salient visual stimuli are perceived automatically, rapidly (<10 ms), in parallel, and at a constant speed regardless of the number of distractors (Treisman, 1985). This phenomenon, known as pre-attentive processing, is key for data visualization, as the right use of visual attributes such as color, contrast, or shape can help us design figures which communicate insights much more effectively (Ware, 2019).

However, many of the visual perception findings are quite specific and niche; to guide effective visualization, we also need some general, high-level rules. One such set of high-level rules are the Gestalt principles of visual perception (see e.g. Cairo, 2012; Ware, 2019;

Rosli and Cabrera, 2015; Vanderplas et al., 2020; Todorovic, 2008; Pinker, 1990). These principles formulate a set of fundamental "laws" related to the perception of (primarily visual) patterns. They were discovered by the founders of the Gestalt school of psychology early in the 20th century, primarily by means of self-observation, and, despite not being grounded in sound understanding of the neural mechanisms of the brain, have nevertheless been later independently supported by findings from cognitive neuroscience (Guberman, 2017; Ware, 2019; Wertheimer, 1938).

Among the Gestalt principles, there are a few which are particularly relevant to data visualization. First, there are the two principles of proximity and similarity. These principles state that objects that which appear in close together and/or similar in appearance tend to be perceived as part of single group. Second, the closure and common region principles state that regions with closed contours tend to be perceived as single, unified objects. Finally, the figure and ground principle states that we tend to perceive one part of a visual scene as the central object (the "figure") and the rest as background (the "ground"). This can be modulated by other Gestalt principle: for example, smaller, darker, and symmetric objects with closed contours are more likely to be perceived as the figure (Ware, 2019).

Whether we intend them to or not, the Gestalt principles affect how our data visualizations are perceived. As such, it is a good idea to try to use them to our advantage. Sometimes, little effort is needed. Such is the case, for example, when identifying clusters in a scatterplot (Rosli and Cabrera, 2015): the principles of proximity and similarity ensure that clouds of similarly shaped and colored points will be perceived as discrete groups. In other cases, however, our design choices can have a significant impact. For instance, another way to make objects be perceived as part of a group is to draw a closed shape around them (Cairo, 2012). Alternatively, if we want to highlight a particular part of a data set, we

can leverage the figure and ground principle and use attributes such as color and saturation to make it stand out (for specific examples, see Franconeri et al., 2021; Wilke, 2019, 34).

Finally, ignoring or violating the Gestalt principles can make our graphics less legible or even misleading. Tufte (2001) had documented many such violations, such as deceptive 3D effects, inadvertent optical illusions caused by dense stripe patterns, and the overuse of superfluous graphical elements termed "chartjunk". Similarly, Cairo (2019) highlights many cases of graphics that violate the Gestalt principles. Finally, there is also the issue of well-designed graphics that communicate no statistical information at all, or even information that is blatantly false (Cairo, 2014; Franconeri et al., 2021; Cairo, 2019; Gelman and Unwin, 2013; Tufte, 2001). Whether intentional or not, poor design can have a significant impact on the legibility and effectiveness of statistical graphics.

2.2 Gestalt and Interactivity

The Gestalt principles apply to both static and interactive visualizations, however, the way they apply is different. Static data visualizations can be designed well or poorly, but this design quality does not change over time. In contrast, interactive visualizations change over time, with user interaction. As such, an interactive visualization may conform to the Gestalt principles at one instance of time and violate them in another. In fact, interactive visualizations can even differ in how well they leverage design principles *across* time: the animations that interactions induce may themselves conform to good or poor design (see Hullman et al., 2013; Ware, 2019).

Thus, interactivity inherently places additional constraints the design of our figures. If we want well-designed interactive figures, we need to ensure that they conform to good design principles across all states induced by user-interaction. This may not always be easy.

We will investigate the design challenged posed by interaction in the following section, on the example of linked selection.

2.3 Case study: Linked Selection

Linked selection, also known as linked brushing or linked highlighting, has been consistently ranked as one of the most useful interactive features in data visualization (see e.g. Buja et al., 1996; Heer and Shneiderman, 2012; Wilhelm, 2003; Ware, 2019; Ward et al., 2015). By clicking or clicking-and-dragging over objects in one plot, users can highlight parts of objects corresponding to the selected cases across all other "linked" plots. This allows the user to rapidly "drill-down" (Dix and Ellis, 1998; Theus, 2002) and explore trends across different dynamically-generated subsets of the data (similar to interactive filtering, see e.g. Heer and Shneiderman, 2012). The ability to quickly materialize different views of the data makes linked selection a versatile tool for data exploration.

Yet, despite its usefulness, linked selection also imposes some fundamental constraints on the figure. Specifically, all objects we visualize need to be able to:

1. Be selected
2. Represent selection
3. (both in a visually and statistically sound way).

While these constraints may not seem too restrictive, we argue that they pose significant challenges for certain visualization styles. Specifically, some kinds of plots may produce undesirable visual behavior when combined with linked selection, and this may in fact explain why stacking is preferred over dodging within the interactive data visualization community. Let's explore this idea in more depth by comparing linked selection with a stacked vs. dodged barplot.

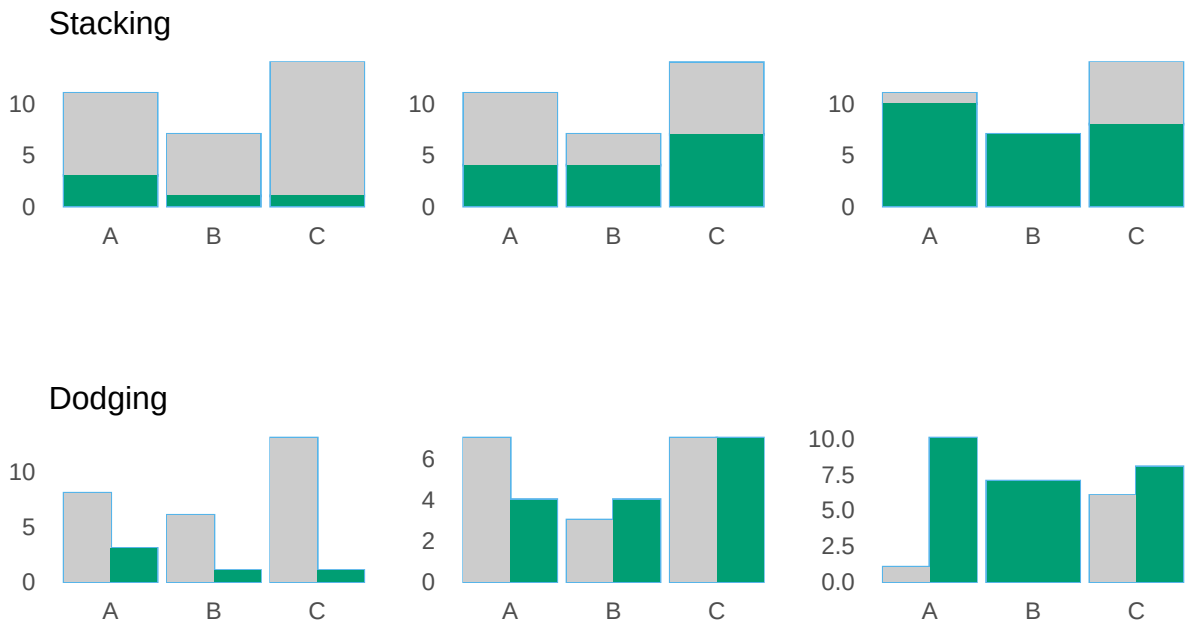


Figure 2: The visual coherence of stacking vs. dodging during linked selection. Dark segments represent selected cases, light segments represent unselected cases, and the plots' contour is shown via a light outline. Plots left to right show more cases being selected. Top row: the contour of a stacked barplot remains constant throughout selection (three bars of the same height). Bottom row: the contour of a dodged barplot changes dramatically with selection. Note the changes to the upper y-axis limit and the absence of the lighter segment among the bar representing category B in the rightmost plot (the darker segment has been stretched to fill the available space; `ggplot2` default for handling empty subcategories). The source of the underlying data is the `mtcars` dataset (Henderson and Velleman, 1981).

First, we propose that stacking is the superior method for *displaying* selection (see Figure 2). Thanks to the four Gestalt principles of proximity, similarity, closure, and common region, in a stacked barplot, each bar presents itself as a single visual entity. In other words, since the stacked segments are placed right on top of each other, and share width and a closed border, they are perceived as part of a unified whole (the bar). This remains true throughout selection - no matter how the heights of the highlighted segments change, the height of the whole stacked bar remains constant, and so does the overall outline of the plot.

Dodged barplots lack this degree of visual consistency. In a dodged barplot, bar segments placed side-by-side in clusters, and, since each segment can be of different length, these clusters lack a shared border (common region). This leads to several design challenges. First, when selection occurs, the jagged contour of the dodged barplot can change dramatically as cases are added or removed. This creates distracting visual clutter that makes it difficult to track changes over time. Second, since the heights of the bar segments are not bound by any value present in the plot (unlike the heights of the segments in the stacked barplot, which are bound by the height of the tallest bar), it is not clear how to handle the upper y-axis limit. On one hand, we could make the upper y-axis limit dynamic and have it change in response to selection. If we do this, we ensure that the bar segments always fit the plotting region, however, we also lose the context that the limit provides each time selection occurs. Alternatively, we can set the limit to a static value, however, then we are at the risk of the bar segments growing outside of the bounds of the plotting region.

Stacked bars may offer several other advantages over dodged bars within the context of linked selection. First, we propose that stacking also provides a simpler mental model for *selection*. Since each stacked bar is a single visual entity, it makes intuitive sense

that clicking it will select the entire corresponding category. Indeed, this is how many interactive data visualization systems implement default linked selection: clicking on part of an object selects all cases corresponding to the whole object (see e.g. Forbes et al., 2023; The Vega Project, 2022; Vanderplas et al., 2018; Xie et al., 2014; Theus, 2002; Urbanek and Theus, 2003; Urbanek, 2011; Swayne et al., 2003). On the other hand, with dodged bars, since the segments do not form a unified whole, the expectation is less clear: when we click on a segment in a dodged barplot, do we select the entire category or just the cases corresponding to the individual segment? Second, stacked barplots maintain constant bar widths and inter-bar gaps throughout selection. In a dodged barplot, it is not clear what to do when some selection groups are empty. Should we leave gaps for the (absent) highlighted segments, leading to asymmetric gaps and potential illusory clustering? Or should we draw wide segments and only shrink them in response to selection, avoiding asymmetry but violating the principle of area proportionality (Cleveland, 1985)? Finally, by presenting fewer visual entities on the screen, stacked bars may also reduce cognitive load (Sweller et al., 2019; for a non-technical overview, see Knaflic, 2015).

To summarize, stacked bars behave in a fairly graceful and visually pleasing way when combined with selection. Dodged bars, on the other hand, lack a bounded uniform outline, and this makes them behave erratically when combined with selection, hurting the interpretability of the plot. These difference may explain the continued popularity of stacking within the interactive data visualization community.

However, a key question remains: when is stacking appropriate? So far, we have focused on the visual aspect of stacking. But stacking goes beyond aesthetics. This will be the subject of the next section.

Figure 3: The representation of a thing is not the thing itself. Left: The Treachery of Images by René Magritte (1929) makes the point that a painting of a pipe is not a pipe. Right: similarly, an image filled with coloured rectangles is not a barplot, unless the rectangles accurately represent some underlying data.

3 Statistics

3.1 From Data to Representation

Data visualizations are made up of geometric objects such as points, lines, or areas. However, these objects alone hold no meaning. Without context, an image filled with coloured rectangles is just that - an image (see Figure 3). To turn the into a plot, we need to make sure that each rectangle accurately represents some underlying data.

Thus, we always have to start with the data. But the raw data is rarely something we can just "throw" at the computer screen. Instead, we often first need to massage and translate it into a format that can be understood by the graphics device. In most data visualization packages, this is done in two steps: aggregation and scaling.

First, the data needs to be aggregated or "wrangled" into sets of statistical summaries. These summaries may be very simple: for instance, in a typical scatterplot, we assign each

point the raw values of the x- and y-variables (the summary function is then just identity, $f(x) = x$). More often, however, the data is grouped and aggregated in some way. For example, as was discussed previously, in a barplot, data points are grouped by the levels of the x-axis variable, and each group is then summarized by some descriptive statistic, such as count, sum, mean, or maximum. Likewise, in a typical histogram, the x-axis variable is divided into bins, and each bin is then summarized by the number of cases that fall inside it.

Second, the computed statistics have to be encoded into graphical attributes such as position, length, size, or colour (see e.g. Franconeri et al., 2021; Wilkinson, 2012; Pinker, 1990). This is the job of specialized functions known as scales or coordinate systems (see e.g. Murrell, 2005, 2007; Wickham, 2010, 2016; Wilkinson, 2012). For instance, in a typical scatterplot, the values of the x-axis variable are mapped to the x-position, such that points corresponding to small values of the x-axis variable are placed near the left-hand side of the plot (and larger values are place near the right-hand side). These mappings may be subject to non-linear transformations, such as log or square-root, and may also translate discrete values, such as barplot categories. Scales and coordinate systems are a rich subject and we have barely scratched the surface here; interested reader is advised to see Wilkinson (2012), Ziemkiewicz and Kosara (2009), or Hotz et al. (2020).

3.2 Combining Statistics

> "This system cannot produce a meaningless graphic, however. This is a strong claim, vulnerable to a single counter-example. It is a claim based on the formal rules of the system, however, not on the evaluation of specific graphics it may produce."

> "Some of the combinations of graphs and statistical methods may be degenerate or bizarre, but there is no moral reason to restrict them."
>
> Wilkinson (2012), The Grammar of Graphics, pp. 15 and 112.

But what does stacking mean in the context of the statistics we are trying to represent? To a data visualization novice, stacking may appear as a purely graphical operation. If they attempt to implement a data visualization pipeline from scratch, they may leave stacking to the very end, after the statistical summaries have already been translated into screen coordinates.

The view of stacking as a purely graphical operation has its appeal. Many modern data visualization systems follow the grammar-based model of visualization (McNutt, 2022; Kim et al., 2022; Vanderplas et al., 2020; Wickham, 2010; Satyanarayan et al., 2014, 2016; Wilkinson, 2012). This model emphasizes compositionality, and enables the user to build plots from independent, modular components. For example, in the popular `ggplot2` library (Wickham, 2010), plots are built by combining components such as geometric objects (called `geoms`) and statistical summaries (`stats`). This grammar-based approach offers a great deal of flexibility and expressiveness, and has gained widespread popularity as a result.

Thus, it may be tempting to treat stacking just as another modular component of a visualization, such that we may be able to freely stack any geometric objects and statistics we like. Unfortunately, there is a deep flaw in this approach. Many data visualization experts have warned about this:

> "Stacking is useful when the sum of the amounts represented by the individual stacked bars is in itself a meaningful amount" (Wilke, 2019, 52).

> "Because this gives the visual impression of one element that is the sum of several others, it is very important that if the element's size is used to display a statistic, then that statistic must be summable. Stacking bars that represent counts, sums, or percentages are fine, but a stacked bar chart where bars show average values is generally meaningless." (Wills, 2011, 112).

> "[...] We do this to ensure that aggregate statistics are always computed over the input data, and so users do not inadvertantly compute e.g., averages of averages, which can easily lead to misinterpretation." (Wu, 2022)

The message is clear: stacking some statistics such as counts, sums, and percentages is acceptable, but stacking others is not. As outlined by Wills, while the sum of grouped counts produces a valid overall count, the sum of group means does not yield a meaningful statistic. Even averaging group means, as noted by Wu (2022), does not produce a valid summary statistic - the mean of group means differs from the grand mean (unless by coincidence). This problem is general and applies beyond barplots; for example, concerns about the statistical validity of stacked density plots have also been recently raised (Pu and Kay, 2020).

What exactly makes stacking counts and sums acceptable but stacking other statistics such as means not? Wilke, Wills, and Wu et al. all argue that the aggregated value should be "meaningful". In the case of sums, there is perhaps a colloquial way to put this: "the whole is equal to the sum of its parts". Or, more specifically:

> "the *sum* of the *sums* on the parts is equal to the *sum* of all values"

But what if we replace word "sum" with some placeholder (such as "foo"):

> "the *foo* of *foos* on the parts is equal to the *foo* of all values"

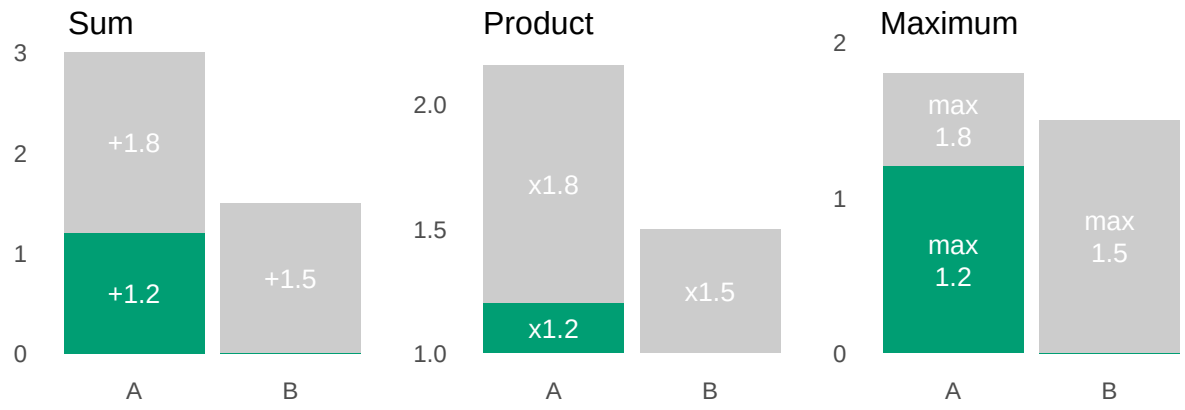


Figure 4: Different operators can be used for stacking. The values `1.2`, `1.8`, `NA`, and `1.5` were assigned to a 2-by-2 table and stacked using three different operators. Left: stacking using the sum operator. Middle: stacking using the product operator (note that the y-axis starts at 1). Right: stacking using the max operator.

Then, we find there are other mathematical operators that satisfy this condition. For example, the product of products or the maximum of maximums are also valid overall statistics, see Figure 4. In fact, there is an entire class of mathematical objects with this exact behavior: monoids. We will explore monoids in the following section.

3.3 Monoids

Monoids are a simple yet powerful concept that plays a key role in several areas of mathematics, including abstract algebra and category theory. The short treatment here draws primarily from Fong and Spivak (2019), Lawvere and Schanuel (2009), Baez (2023), and Milewski (2018). For a particularly accessible introduction, readers are encouraged to seek out Fong and Spivak (2019) or Lawvere and Schanuel (2009).

While the term "monoid" may sound intimidating, the concept is really quite simple. A monoid represents a "whole equal to the sum of its parts", if we relax our notion of what

it means to "sum". More formally, a monoid is a tuple $(M, e, \otimes)$ consisting of:

a. A collection (set) of objects M

b. A neutral element $e \in M$ called the *monoidal unit*

c. A binary operation $\otimes : M \times M \to M$ called the *monoidal product*

Conforming to two rules:

1. *Unitality*: $x \otimes e = e \otimes x = x$
2. *Associativity*: $x \otimes (y \otimes z) = (x \otimes y) \otimes z = x \otimes y \otimes z$

(for all $x, y, z \in M$)

In plain words, when we have a monoid, we have some set of elements ($m \in M$), and a way to combine these elements ($\otimes$), such that the order in which we do the combining does not matter (associativity). Further, among the elements, we also have a special element (e), called the "neutral" or "identity" element, that, when combined with any other element, does nothing. This identity element is always unique (the proof is straightforward, for reference see e.g. Fong and Spivak, 2019; Lawvere and Schanuel, 2009).

Let's return to the book example from Section 1. Does it represent a monoid? First of all, what is our set M? This could be the set of all book stacks we can form by combining books in our bookshelf, including an "empty" stack of zero books, "short" stacks made up of a single book each, and the tallest stack formed by taking all of the books in the bookshelf. Second, do we have a neutral element? Sure, this is the empty stack of zero books. Third, what is our binary operation? This is just the operation of taking one stack of books and putting it on top of another.

Now we just need to verify that our example conforms to the two rules. First, is the operation associative? Yes, since a stack of detective novels will be the same height

regardless of whether we put the novels by Agatha Christie on top of Jo Nesbø books or vice versa. Second, is the operation unital? Also yes, since adding zero books on top of a stack does not change its height. Therefore, we do have a monoid.

We can describe the book example a bit more formally. Specifically, summation on natural numbers $(\mathbb{N}, 0, +)$ is a typical example of a monoid, because, as we have shown, it is associative and unital. Other typical examples of monoids include the previously mentioned products of real numbers $(\mathbb{R}, 1, \times)$, and the min and max operators $(\mathbb{R}, \infty, \min)$ and $(\mathbb{R}, -\infty, \max)$. As a counterexample, exponentiation is not a monoid, since it is not associative: $x^{(y^z)} \neq (x^y)^z$.

The definition of a monoid is quite broad however, and encompasses more exotic structures than just operations on numbers. For example, the set of booleans $\mathbb{B} = \{True, False\}$ equipped with either of the logical operators **AND** or **OR** is also a monoid, and so is the multiplication of $n \times n$ square matrices. Likewise, the operation of concatenating vectors and computing the euclidean distance is also a monoid (Stepanov and McJones, 2009):

$$||\big(||(x, y)||_2, z\big)||_2 = \sqrt{\left(\sqrt{x^2 + y^2}\right)^2 + z^2} = \sqrt{x^2 + y^2 + z^2} = ||(x, y, z)||_2$$

Even very "non-number-like" things such as strings can be a monoid:

$$\texttt{"hello"} + \texttt{""} = \texttt{""} + \texttt{"hello"} = \texttt{"hello"}$$

$$(\texttt{"quick"} + \texttt{"brown"}) + \texttt{"fox"} = \texttt{"quick"} + (\texttt{"brown"} + \texttt{"fox"}) = \texttt{"quick brown fox"}$$

3.4 Groups

Groups are another fundamental concept in mathematics, representing the idea of symmetry and reversible transformations. While introducing two algebraic concepts in quick

succession may seem like a big ask, a group is essentially just a monoid with one additional property. Specifically, for all $x, y, z \in M$, we have:

3. *Inverse operator*: There exists $\otimes^{-1}$ such that if $x \otimes y = z$ then $z \otimes^{-1} x = y$

In simple terms, a group embodies the idea that "the whole is equal to the sum of its parts, *in a non-destructive way*". That is, when we combine two elements, we can always recover either one by "subtracting" the contribution of the other. Of note, like the identity element for monoids, the inverse operator in a group is also always unique (see e.g. Fong and Spivak, 2019; Lawvere and Schanuel, 2009).

Turning back to the book example, we can remove Agatha Christie novels off the detective novel stack and recover the height of the rest of the books. Thus, summation on natural numbers is a group: $2 + 3 = 5 \implies 5 - 2 = 3$ (technically, we now have to identify the underlying set as the set of integers $\mathbb{Z}$). But, for example, the min and max operators lack an inverse and so do not constitute groups: if $\max(x, 6) = 6$, then there is no way to recover x from the combined result (6) alone.

3.5 Groups, Monoids, and Stacking

We now arrive at the core idea of the present paper. Suppose that we have some data set D, a way of summarizing this data set via an associative binary operation $\otimes$, and a neutral value with respect to this operation e. In other words, $(D, \otimes, e)$ forms a monoid. Define $F(A)$ as the result of summarizing or "folding" some subset $A \subseteq D$ using the binary operation:

$$F(A) = a_1 \otimes a_2 \otimes a_3 \ldots \otimes a_n$$

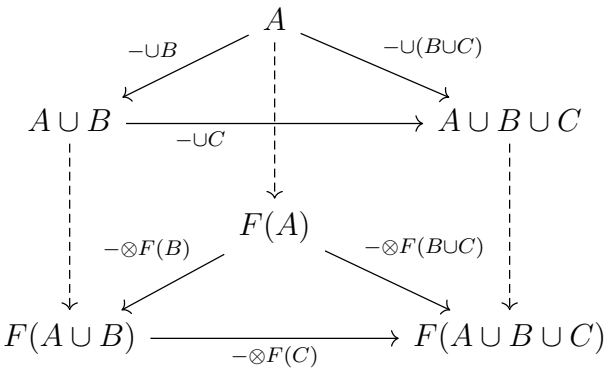
Note that we could have written this as $F(A) = (a_1 \otimes (a_2 \otimes (a_3 \otimes (\ldots \otimes a_n))))$, however, since the operation is associative, we do not have to worry about brackets. Also, if A is empty, then clearly:

$$F(\varnothing) = e$$

Now comes the key concept: if the conditions above hold, then combining summaries of two disjoint subsets of the data set gives us the same result as summarizing the union of the original subsets:

$$\begin{aligned} F(A) \otimes F(B) &= (a_1 \otimes a_2 \otimes \ldots a_n) \otimes (b_1 \otimes b_2 \otimes \ldots b_n) \\ &= a_1 \otimes a_2 \otimes \ldots \otimes a_n \otimes b_1 \otimes b_2 \otimes \ldots \otimes b_n \qquad \text{(by associativity)} \\ &= F(A \cup B) \end{aligned}$$

Another way to show this is the following commutative diagram:



The top and bottom triangles in the diagram both show associativity: in the top triangle, we can either first take the union of A and B and then union the result with C, or immediately union A with $B \cup C$, and the result will be identical. The same applies for the monoidal summary in the bottom triangle. By saying that the diagram commutes, we mean that any two parallel paths (the same start and endpoint) always produce the same

result. Practically, this means we can either first summarize each set and then combine the summaries, first union the sets and then summarize the union, or do some mix of both, and we always end up with a valid summary. As a sidenote, the diagram above actually corresponds to something called a functor or monoid homomorphism, however, that is beyond the scope of the present paper (interested reader should see Fong and Spivak, 2019; Lawvere and Schanuel, 2009).

But what we have now is precisely the idea of a stackable statistic. If our statistic conforms to the diagram, then the combined summary of subsets of the data is always a valid summary of the whole data set. In Wilke's words, we have discovered what makes the combined summary a "meaningful amount"!

It might seem that we are done. However, there is still one subtle issue left. Suppose we summarize three parts of the data A, B, and C, and combine these summaries by repeatedly applying the operation, as a kind of running total F^*:

$$\begin{aligned} F^*\big(F(A), F(B), F(C)\big) &= \big(F(A), F(A) \otimes F(B), F(A) \otimes F(B) \otimes F(C)\big) \\ &= \big(F(A), F(A \cup B), F(A \cup B \cup C)\big) \end{aligned}$$

Notice that we end up with summaries on nested subsets of the data, i.e. $F(A)$, $F(A \cup B)$, and $F(A \cup B \cup C)$. This is fine, for example, when we do linked selection with a single highlighted group and want to compare the summary on the highlighted subset (A) vs. the entirety the data ($A \cup B$). The resulting statistics give us what we need.

However, what if we want to compare the subsets A and B directly? For example, suppose we do linked selection and with two highlighted groups (A and B). Unfortunately, the properties of monoids do not guarantee that we will be able to recover $F(B)$ from $F(A \cup B)$ using $F(A)$. Put plainly, the monoidal operation may "collapse" the information contained in the disjoint parts, see Figure 5. To ensure that $F(B)$ is recoverable from

$F(A \cup B)$, we also need the inverse operator $\otimes^{-1}$:

$$F(B) = F(A \cup B) \otimes^{-1} F(A)$$

Importantly, even when the operation does not have an inverse, $F(A) \otimes F(B)$ is still a valid summary of $A \cup B$. It is just that $F(A) \otimes F(B)$ may "forget" the information contained in either of its two constituent parts. Note also that this is not a problem of different permutations of parts producing different sequences of accumulated summaries - this will happen regardless of whether there is an inverse or not.

To summarize, whether our statistic is a monoid or a group determines what kinds of comparisons we can make with stacked objects:

- Monoids: comparing nested subsets, part vs. whole comparisons
- Groups: comparing disjoint subsets, part vs. part comparisons

This becomes important when implementing interactive data visualization features such as linked selection. Specifically, if we want to represent a single highlighted group vs. the rest of the data via stacked objects, then it's enough for our summary statistic to be a monoid. If we want to represent multiple highlighted groups, then our statistic needs to have an inverse as well. If we fail either of these conditions, we may produce representations which are either inaccurate (in the case of non-monoidal statistics), or accurate but not "full" (in the case of monoids but no inverse).

3.6 Groups, Monoids, and Graphics

We can use the connection between summary statistics and algebraic structures to offload some of the cognitive effort required when thinking about new types of interactive visualizations. For example, while not generally thought of as a summary statistic, the convex

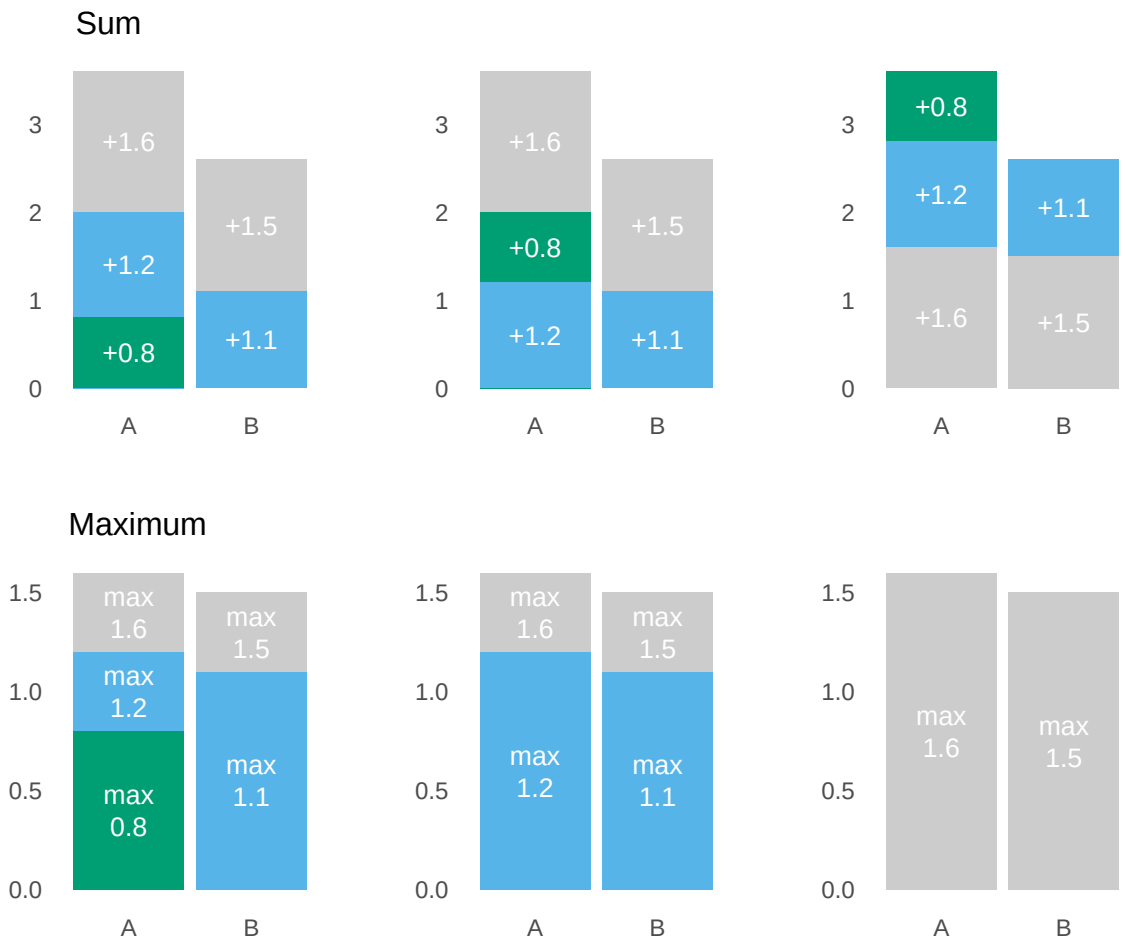


Figure 5: Monoids can "collapse" information about disjoint subsets. Top: since sum is a group with an inverse (minus), we can recover the value of each part by subtracting the heights of the preceding segments (i.e. the part order does not matter). Bottom: since maximum lacks an inverse, we may lose information about the individual parts under certain permutations (note the absence of segments in the middle and rightmost plot).

hull (see e.g. Barber et al., 1996) provides a way to reduce a set of data points into a simpler "summary", see Figure 6. We can recast convex hull algorithm as a monoid: starting with an empty set, we iterate through the data, adding points if they lie outside of the current hull (and removing any points in the interior of the new hull). This may not be the most computationally efficient method of finding convex hulls, however, since it does meet the definition of a monoid, this tells us *what convex hulls can be used for*. If we combine two convex hulls, we know that the new hull will be a valid summary for the union of the underlying data points. In turn, this allows us to implement interactive features such as single-group highlighting/selection, safe in the knowledge that the underlying structure is preserved.

However, does the operation of adding a point to a convex hull have an inverse? No. Once we add a point to the hull, we cannot recover the old hull by "subtracting" that point - we do not know how many points from the interior of the old hull we removed. This tells us that merging two convex hulls may irrevocably collapse information. As such, we cannot safely use convex hulls to display multiple selection groups, without resorting to some ad-hoc methods.

While it is possible to reach the same conclusions via ad hoc reasoning or manual testing, these methods may be time-consuming and have to be applied to each new visualization style separately. By grounding our understanding in the simple theory of groups and monoids, we can bypass most of this effort and reason about the properties of the visualization directly.

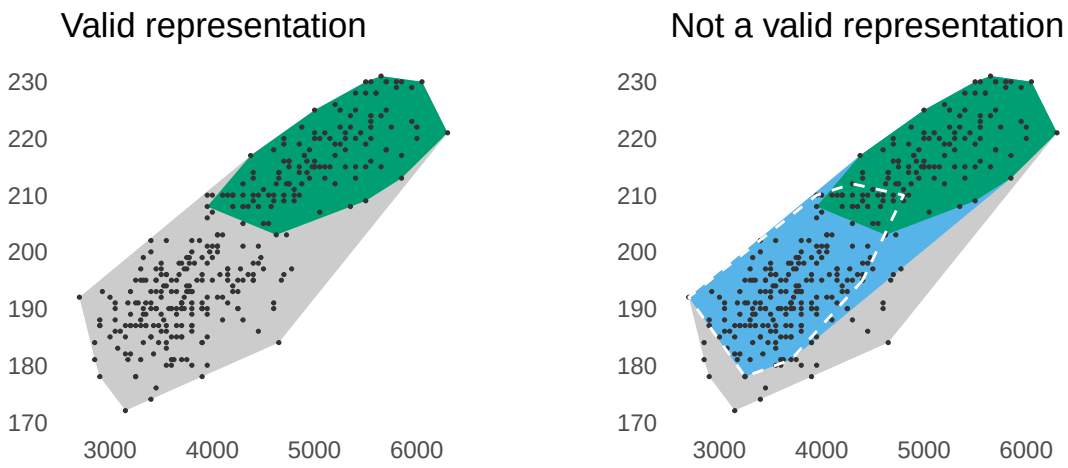


Figure 6: Convex hull can be used as a valid representation of single-group selection but not multi-group selection. The light polygons shows the convex hull of the whole dataset, the dark polygon shows the hull of one selection group, the white dashed line shows the hull of a second selection group, and the medium-brightness polygon shows the combined hull of the two selection groups. Points show individual cases. Left: since convex hull is a monoid, we can use it to show a single highlighted group. Right: because convex hull lacks an inverse, after combining two hulls, we cannot guarantee that we will be able to recover either of the original hulls - we do not know how many interior points were "forgotten". The source of the underlying data is the Palmer penguins data set (Horst et al., 2020).

3.7 Groups, Monoids, and the Neutral Element

There are other benefits that groups and monoids offer for visualizing data. One of these is the existence of the neutral element e (see Section 3.3). While seemingly inconspicuous, the neutral element can actually be quite useful.

Specifically, the neutral element ensures that we can always represent empty subsets of our data. Turning once more to the book example from Section 1, suppose we split the books into stacks based on the book's genre and the author's gender, and then summarize each stack by its total and average word-count. What if there are no books by male sci-fi authors? For total word-count, we have a meaningful default value - zero - as in, there are zero words across the zero books by male sci-fi authors. However, what about the average word-count? Here we run into trouble, since the mean of an empty set is not defined. It simply does not make sense to talk about the "average word-count across zero books".

A common approach is to omit the objects representing empty subsets. However, with non-monoidal summaries, this approach is flawed and leads to ambiguity. Suppose we are drawing the bars of averages. If we omit bars representing empty categories, then the missingness of a bar can indicate one of two things: either there are *no* cases in the category, or there *are* cases and their average is equal to the lower y-axis limit. There is just no way to tell from the graphic alone. In contrast, with monoidal summaries such as sums, the absence of an object (bar) sends a clear message. Regardless of whether there are zero or hundred cases in a category, if the corresponding bar is missing, then we know their sum is zero.

The neutral element can also provide a new perspective on the longstanding debate in data visualization: should the base of a barplot always start at zero? Some authors argue that starting from a non-zero base renders bar lengths meaningless (Cleveland, 1985). On

the other hand, Wilkinson (2012), for example, suggests that the bar base is really a statement about scales rather than graphics, and as such it does not have to be zero. We offer the following new perspective: often, it might make sense to start the base of the barplot at e, the neutral element, since this represents the default state of "doing nothing". In other words, the bar base is tied to the properties of the *statistic*, rather than just the graphic or the scale. For instance, in a logarithmic barplot, the underlying monoidal operation is multiplication, and so the base of the bars should start at one.

These example provides further support for the claim that we cannot separate statistics from other components such as graphics and scales. Even before we apply any transformations, the method we have used to derive our summary statistics determines important properties of our visualization. We need to be mindful of this.

3.8 Groups, Monoids, and Computation

Groups and monoids also offer some inherent computational advantages. Specifically, if a summary statistic is a monoid, associativity guarantees that it can be computed in a single pass through the data (i.e. $O(n)$ complexity), and in a distributed/parallel fashion (see e.g. Lämmel, 2008; Lin, 2013). This is makes monoids popular in functional programming, where they are known as the `fold` or `reduce` functions (see e.g. Milewski, 2018; Abelson and Sussman, 2022).

While computational efficiency is usually not the number one concern in static visualization, it becomes significant in the context of interactive graphics. Here, we may be able to use the properties of groups and monoids to avoid unnecessary computation. Specifically, when using groups and monoids, we can exploit associativity: the fact that summaries on fine-grained subsets of the data "add up" to the summaries on coarse-grained subsets.

Consider a stacked histogram, with linked selection, reactive axes, and interactive manipulation of binwidth and anchor. When the user engages in selection, the counts within the individual highlighted segments may change but the counts within the whole bars will stay constant. Therefore, there is no need to update the upper y-axis limit. However, if the user updates either the binwidth or the anchor, then both the counts within the whole bars *and* within the highlighted segments may change, and so we need to update the graphic appropriately.

In other words, associativity allows us to implement strategic updating. Fine-grained summaries (individual segments) need to be updated whenever coarse-grained summaries (whole stacked bars) change, but not vice versa. By pre-computing summaries on different levels of aggregation, we may be able to reduce computational effort.

In fact, if our summary statistic has an inverse, we can be even more computationally efficient. Suppose that one data point out of a thousand is removed from selection. If our summary has an inverse, we can use it to subtract the contribution of that single data point, instead of having to recompute the whole summary from scratch. This is, in fact, the principle behind the fast updates in the popular JavaScript library Crossfilter, which is unfortunately no longer actively maintained (Crossfilter Organization, 2023; see also the Github discussion initiated by Monfera, 2017)[1].

However, while the presence of the inverse does bring some advantages, it is also the case that fewer summary statistics have an inverse. For instance, as was mentioned before, while maximum operator is a valid monoid, it lacks an inverse and as such does not meet the definition of a group. Once we summarize some data with the maximum, we cannot retrieve the second or third highest value - that information is gone forever. And there are

[1]Note that, as the present date, we are not aware of any explicit discussion of monoids or groups in the Crossfilter documentation

many other operations (such as the convex hull in Figure 6) which lack an inverse. As such, when designing and interactive data visualization library, we need to carefully weigh the ability to support different kinds of summary statistics (general, monoids, groups) against the ability to support wide range of interactions (e.g. single-group selection vs. multi-group selection).

4 Discussion

We hope we have demonstrated that, when designing interactive graphics, algebraic thinking is invaluable. To support certain interactive features such as linked selection, the statistics underlying our plots must possess appropriate algebraic properties. If we want to display the result of splitting our data into parts and then combining these parts back together, then we need either monoids or groups. Monoids are for comparing nested subsets of our data (single-group linked selection) whereas groups are for comparing disjoint parts of our data (multi-group linked selection). The properties of these structures ensure that our figures will be statistically sound, computationally efficient, and adhere to good design principles.

We do not advocate for exclusive use of groups and monoids. Rather, we want to highlight the inherent trade-off present in the choice of a summary statistic. Few summary statistics meet the definition of a group, but, if we choose such a statistic, then we can do a lot of things with it. With a monoid, our options are more limited, and with a non-monoidal statistic, even more so. Like many past authors (such as Tufte, 2001; Tukey et al., 1977; Wilkinson, 2012), we want to emphasize the importance of thinking critically about the mathematical structure underlying our graphics. Geometric objects, statistics, and interaction are all deeply connected, and groups and monoids merely allow us to carve

out a small space of fairly well-behaved visualizations.

On that note, while we have focused on unitality, associativity, and inverses, it may be useful to consider other algebraic properties as well. For instance, many of the visual attributes that we use to represent our data, such as width, height, or area, can only *grow* in one direction (there is no such thing as "negative area"). If this is the case, then our summary statistic should be *monotonic*, such that if $x_1 \leq y_1$ and and $x_2 \leq y_2$ then $x_1 \otimes x_2 \leq y_1 \otimes y_2$. Similarly, in many types of plots, the order of the data points does not matter, however, in some, such as time-series plots, it does. Thus, it may be important to determine whether or not the statistics underlying our plots are *commutative*, such that $x \otimes y = y \otimes x$. Further exploration of these properties in relation to interactive graphics may yield useful insights, however, that is outside of the scope of the present paper.

On a more practical note, interactive graphics require efficient pipelines for transforming raw data into summary statistics, and, currently, we still do not possess a formal framework for doing this. A decade ago, Wickham et al. (2009) noted the lack of such data pipelines, and, despite some attempts at implementation since then (see e.g. Lawrence et al., 2009), the same sentiment was echoed again recently (Vanderplas et al., 2020). The ultimate goal is to minimize recomputation during user interaction, and this is where the computational shortcuts that groups and monoids provide may prove invaluable. We envision a visualization pipeline structured as a hierarchy of reactive partitions, where only the downstream (fine) partitions need to be recomputed when the upstream (coarse) partitions change, but not vice versa. We are currently exploring this idea in an R package called `plotscaper`[2] (note that the project is still under development).

Data visualization, in practice, is rarely about mathematical purity. Real world data is often flawed or incomplete, and it takes cunning and experience to dredge up valuable

[2] `https://github.com/bartonicek/plotscaper`

insights from it. Nevertheless, we believe that simple algebraic thinking offers a valuable foundation. Concepts like groups and monoids provide a solid framework for reasoning about graphics, interactive or otherwise. By grounding our thinking in these concepts, we may be able to offload some of the required cognitive effort and envision new types of robust and expressive interactive graphics.

5 Disclosure Statement

The authors report there are no competing interests to declare.

Bibliography

Abelson, H. and Sussman, G. J. (2022). *Structure and Interpretation of Computer Programs: JavaScript Edition*. MIT Press.

Baez, J. (2023). Applied Category Theory Course. [Online; accessed 3. Nov. 2023].

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Bertin, J. (1983). *Semiology of graphics*. University of Wisconsin press.

Buja, A., Cook, D., and Swayne, D. F. (1996). Interactive high-dimensional data visualization. *Journal of computational and graphical statistics*, 5(1):78–99.

Byron, L. and Wattenberg, M. (2008). Stacked graphs–geometry & aesthetics. *IEEE transactions on visualization and computer graphics*, 14(6):1245–1252.

Cairo, A. (2012). *The Functional Art: An introduction to information graphics and visualization*. New Riders.

Cairo, A. (2014). Graphics lies, misleading visuals: Reflections on the challenges and pitfalls of evidence-driven visual communication. In *New challenges for data design*, pages 103–116. Springer.

Cairo, A. (2019). *How charts lie: Getting smarter about visual information*. WW Norton & Company.

Cleveland, W. S. (1985). *The elements of graphing data*. Wadsworth Publ. Co.

Cleveland, W. S. and McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554.

Crossfilter Organization (2023). Crossfilter. [Online; accessed 31. Oct. 2023].

Dix, A. and Ellis, G. (1998). Starting simple: adding value to static visualisation through simple interaction. In *AVI '98: Proceedings of the working conference on Advanced visual interfaces*, pages 124–134. Association for Computing Machinery, New York, NY, USA.

Fong, B. and Spivak, D. I. (2019). *An invitation to applied category theory: seven sketches in compositionality*. Cambridge University Press.

Forbes, J., Cook, D., Ebert, A., Hofmann, H., Hyndman, R., Lumley, T., Marwick, B., Sievert, C., Sun, M., Talagala, D., Tierney, N., Tomasetti, N., Wang, E., and Zhou, F. (2023). *eechidna: Exploring Election and Census Highly Informative Data Nationally for Australia*. https://github.com/jforbes14/eechidna/, https://jforbes14.github.io/eechidna/.

Franconeri, S. L., Padilla, L. M., Shah, P., Zacks, J. M., and Hullman, J. (2021). The science of visual data communication: What works. *Psychological Science in the public interest*, 22(3):110–161.

Gelman, A. and Unwin, A. (2013). Infovis and statistical graphics: different goals, different looks. *Journal of Computational and Graphical Statistics*, 22(1):2–28.

Goebel, R., Muckli, L., and Kim, D.-S. (2004). Visual system. *The Human Nervous System Elsevier, San Diego*, pages 1280–1305.

Guberman, S. (2017). Gestalt theory rearranged: Back to wertheimer. *Frontiers in psychology*, 8:1782.

Heer, J. and Bostock, M. (2010). Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 203–212.

Heer, J. and Shneiderman, B. (2012). Interactive dynamics for visual analysis: A taxonomy of tools that support the fluent and flexible use of visualizations. *Queue*, 10(2):30–55.

Henderson, H. V. and Velleman, P. F. (1981). Building multiple regression models interactively. *Biometrics*, pages 391–411.

Horst, A. M., Hill, A. P., and Gorman, K. B. (2020). *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*. R package version 0.1.0.

Hotz, I., Bujack, R., Garth, C., and Wang, B. (2020). Mathematical foundations in visualization. *Foundations of Data Visualization*, page 87.

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

Kim, H., Rossi, R., Du, F., Koh, E., Guo, S., Hullman, J., and Hoffswell, J. (2022). Cicero:

A declarative grammar for responsive visualization. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Knaflic, C. N. (2015). *Storytelling with data: A data visualization guide for business professionals*. John Wiley & Sons.

Knudsen, E. I. (2020). Evolution of neural processing for visual perception in vertebrates. *Journal of Comparative Neurology*, 528(17):2888–2901.

Kosara, R. (2016). Stacked Bars Are the Worst. [Online; accessed 10. Nov. 2023].

Lämmel, R. (2008). Google's mapreduce programming model—revisited. *Science of computer programming*, 70(1):1–30.

Lawrence, M., Wickham, H., Cook, D., Hofmann, H., and Swayne, D. F. (2009). Extending the ggobi pipeline from r: Rapid prototyping of interactive visualizations. *Computational Statistics*, 24:195–205.

Lawvere, F. W. and Schanuel, S. H. (2009). *Conceptual mathematics: a first introduction to categories*. Cambridge University Press.

Lin, J. (2013). Monoidify! monoids as a design principle for efficient mapreduce algorithms. *arXiv preprint arXiv:1304.7544*.

Magritte, R. (1929). The Treachery of Images (This is Not a Pipe) (La trahison des images [Ceci n'est pas une pipe]) | LACMA Collections. [Online; accessed 5. Jun. 2024].

McNutt, A. M. (2022). No grammar to rule them all: A survey of json-style dsls for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):160–170.

Milewski, B. (2018). *Category theory for programmers*. Blurb.

Monfera, R. (2017). [WIP] Crossfilter discussion · Issue #1316 · plotly/plotly.js. [Online; accessed 22. Feb. 2024].

Murray, S. (2017). *Interactive data visualization for the web: an introduction to designing with D3*. " O'Reilly Media, Inc.".

Murrell, P. (2005). *R graphics*. Chapman and Hall/CRC.

Murrell, P. (2007). grid graphics.

Pinker, S. (1990). A theory of graph comprehension. *Artificial intelligence and the future of testing*.

Playfair, W. (1801). *The commercial and political atlas: representing, by means of stained copper-plate charts, the progress of the commerce, revenues, expenditure and debts of england during the whole of the eighteenth century*. T. Burton.

Playfair, W. (1822). A letter on our agricultural distresses. *Their Causes and Remedies, 1st ed. William Sams, London (1821)*.

Pu, X. and Kay, M. (2020). A probabilistic grammar of graphics. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Quadri, G. J. and Rosen, P. (2021). A survey of perception-based visualization studies by task. *IEEE transactions on visualization and computer graphics*.

Rosli, M. H. W. and Cabrera, A. (2015). Gestalt principles in multimodal data representation. *IEEE computer graphics and applications*, 35(2):80–87.

Satyanarayan, A., Moritz, D., Wongsuphasawat, K., and Heer, J. (2016). Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350.

Satyanarayan, A., Wongsuphasawat, K., and Heer, J. (2014). Declarative interaction design for data visualization. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 669–678.

Sievert, C. (2020). *Interactive web-based data visualization with R, plotly, and shiny*. CRC Press.

Stepanov, A. A. and McJones, P. (2009). *Elements of programming*. Addison-Wesley Professional.

Swayne, D. F., Lang, D. T., Buja, A., and Cook, D. (2003). GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Comput. Statist. Data Anal.*, 43(4):423–444.

Sweller, J., van Merriënboer, J. J., and Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational psychology review*, 31:261–292.

The Vega Project (2022). Example Gallery: Interactive. [Online; accessed 22. Aug. 2022].

Theus, M. (2002). Interactive Data Visualization using Mondrian. *J. Stat. Soft.*, 7:1–9.

Thudt, A., Walny, J., Perin, C., Rajabiyazdi, F., MacDonald, L., Vardeleon, D., Greenberg, S., and Carpendale, S. (2016). Assessing the readability of stacked graphs. In *Proceedings of Graphics Interface Conference (GI)*.

Todorovic, D. (2008). Gestalt principles. *Scholarpedia*, 3(12):5345.

Treisman, A. (1985). Preattentive processing in vision. *Computer vision, graphics, and image processing*, 31(2):156–177.

Tufte, E. R. (2001). *The visual display of quantitative information*. Graphics Press LLC, Cheshire, Connecticut.

Tukey, J. W. et al. (1977). *Exploratory data analysis*, volume 2. Reading, MA.

Unwin, A. (2018). *Graphical data analysis with R*. Chapman and Hall/CRC.

Urbanek, S. (2011). iplots extreme: next-generation interactive graphics design and implementation of modern interactive graphics. *Computational Statistics*, 26(3):381–393.

Urbanek, S. and Theus, M. (2003). iplots: high interaction graphics for r. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*. Citeseer.

U.S. Department of Energy (2024). Fueleconomy.gov - the official u.s. government source for fuel economy information. [Online; accessed 23. Oct. 2024].

Vanderplas, J., Granger, B., Heer, J., Moritz, D., Wongsuphasawat, K., Satyanarayan, A., Lees, E., Timofeev, I., Welsh, B., and Sievert, S. (2018). Altair: interactive statistical visualizations for python. *Journal of open source software*, 3(32):1057.

Vanderplas, S., Cook, D., and Hofmann, H. (2020). Testing statistical charts: What makes a good graph? *Annual Review of Statistics and Its Application*, 7:61–88.

Ward, M. O., Grinstein, G., and Keim, D. (2015). *Interactive Data Visualization: Foundations, Techniques, and Applications*. CRC Press.

Ware, C. (2019). *Information visualization: perception for design*. Morgan Kaufmann.

Wertheimer, M. (1938). Laws of organization in perceptual forms.

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

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H., Lawrence, M., Cook, D., Buja, A., Hofmann, H., and Swayne, D. F. (2009). The plumbing of interactive graphics. *Computational Statistics*, 24:207–215.

Wilhelm, A. (2003). User interaction at various levels of data displays. *Comput. Statist. Data Anal.*, 43(4):471–494.

Wilke, C. O. (2019). *Fundamentals of data visualization: a primer on making informative and compelling figures*. O'Reilly Media.

Wilkinson, L. (2012). *The grammar of graphics*. Springer.

Wills, G. (2008). Linked data views. In *Handbook of data visualization*, number ch. II. 9, pages 217–241. Springer Berlin/Heidelberg, Germany.

Wills, G. (2011). *Visualizing time: Designing graphical representations for statistical data*. Springer Science & Business Media.

Wu, E. (2022). View composition algebra for ad hoc comparison. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2470–2485.

Xie, Y., Hofmann, H., and Cheng, X. (2014). Reactive programming for interactive graphics. *Statistical Science*, pages 201–213.

Ziemkiewicz, C. and Kosara, R. (2009). Embedding information visualization within visual representation. In *Advances in Information and Intelligent Systems*, pages 307–326. Springer.