

Mind Your ν s and Qs: A Categorical Framework for Specifying Visualization Components

Hannah Aizenman *

Computer Science, The Graduate Center, CUNY

Thomas Caswell ‡

National Synchrotron Light Source II, Brookhaven National Lab

Mikael Vejdemo-Johansson †

Mathematics, CUNY College of Staten Island

Michael Grossberg §

Computer Science, City College of New York, CUNY

ABSTRACT

Graphical grammars express how to put visual elements together; we propose a methodology for defining those visual elements in a manner that guarantees that they are structure preserving. Fiber bundles provide a uniform interface for describing a variety of data - tables, images, networks, etc. - in a manner that independently encodes the mathematical structure of the continuity and fields of the dataset. Modeling the data containers that store the datasets as sheaves provides a framework for specifying visualization methods that are designed to work regardless of how the dataset is stored - whether on disk, distributed, or on demand. Specifying the visualization library components as natural transforms of sheaves means that the constraints that the component must satisfy to be structure preserving can be specified as the set of morphisms on the data and graphic sheaves, including the structure on the continuity and fields of the data. Formally expressing how visual elements are constructed using category theory means we can translate those constraints into pseudocode, which can be used to implement the words in the visual sentence in a way that ensures that the sentences generate images that are faithful translations of the data.

1 INTRO

A visualization described by a visual grammar is a faithful translation of the underlying data only if the visual components that generated the graphic are implemented in a structure preserving way. While visualization libraries aim to do so, architecture decisions and ongoing development can lead to bugs. For example, the Matplotlib Python visualization library has different APIs for accessing each type of data (e.g. line and image), which leads to changes to the data (e.g. unit conversions) not always applied consistently across plotting methods. This inconsistency motivated us to develop a framework that provides a uniform interface for arbitrary datasets in arbitrary containers, uses category theory as the language for expressing what structure needs to be preserved across mapping and provides guidance for the construction and testing of structure preserving visualization library components.

2 RELATED WORK

Our model generalizes existing mathematically derived models of visualization by allowing arbitrary structures, explicitly incorporating topology, and by providing an explicit framework for translating the theoretical ideas into buildable components. Bertin [2] codified the idea that properties of data and graphics should match, using the Steven’s measurement scales [22] as a case study. Mackinlay

and Wilkinson translate algebras on data to grammars for specifying graphics to develop frameworks for designing structure preserving visualization tools [15, 14, 24], while our framework uses the same abstractions for data and graphics. Vickers et al [23] apply category theory to visualization semantics, while our focus is structural. Bartonicek built a tool based on the algebraic properties of summary statistics [1]; our work provides a more general framework for codifying those properties and proposes that sheaves are a better abstraction for reasoning about connectivity constraints. Similarly to Kindlemann and Scheidegger’s algebraic visualization design (AVD) framework [10] — but with a focus on software architecture rather than visualization design — we present a uniform abstract data representation layer for ensuring that the visualization should not change if the data representation changes, define the conditions under which data is mapped unambiguously to visual encodings [25], and provide a methodology for verifying that changes in data should correspond to changes in the visualization.

3 UNIFORM ABSTRACTION FOR DATA & GRAPHICS

Generally, preserving structure means that a visualization is expected to preserve the field properties and connectivity of the corresponding dataset. We use **field**¹ to denote the set of values of the same type, e.g. date, location, stations, and precipitation in a weather station dataset and **connectivity** to denote how elements are arranged in a dataset [24], e.g. as 0D (•) points, along a 1D (—) line, in a 2D (■) surface, as a 3D trajectory, a volumetric cube, or as a network. We propose a mathematical abstraction of the data input and pre-rendered graphic output that allows us to express the structure of data and graphics using the type signature `dataset : connectivity → fields`, which means visualization algorithms can check the data input against a signature to ensure that the assumptions of the data match the assumptions of the algorithm, that output fields are equivalent to the input fields, and that the connectivity of the output graphic matches the connectivity of the input data.

3.1 Abstract Data Representation

We model data using a structure from algebraic topology called a fiber bundle that can encode topological properties, field types, and data values in a uniform manner. We extend Butler’s proposal of using bundles as an abstraction for visualization data [5, 4] by incorporating Spivak’s methodology for encoding named data types from his fiber bundle representation of relational databases [20, 21]. We propose that the **total space** E of a bundle can encode the mathematical space in which a dataset is embedded, the **base space** K can encode the connectivity of the dataset, the **fiber space** F can encode the field properties of the dataset, and that the datasets can be encoded as **section functions** τ from the base space to the fiber space.

The base space K can be thought of as an indexing set for the data because the index (usually) encodes the connectivity. Framing this

¹In this paper, color in definitions, equations, and visualizations is used to group conceptually related terms [8], e.g. **field** and **fiber**.

*email: haizenman@gradcenter.cuny.edu

†e-mail: mvj@math.csi.cuny.edu

‡e-mail: tcaswell@bnl.gov

§e-mail: mgrossberg@ccny.cuny.edu

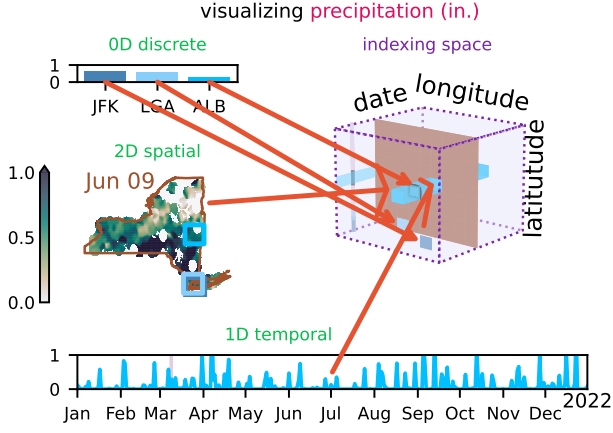


Figure 1: This weather station data has multiple embedded continuities - points at each time and position, timeseries at each position, and maps at each time. The corresponding visualizations - bar chart, timeseries, and map - each preserve the continuity of the subset of the data they visualize by not introducing or leaving out values and preserving the relative positioning of continuous values.

in the language of topology provides us with an abstraction of our data that can be unified over data representation (in memory, distributed, streaming) and over different connectivity types (discrete, d-dimensional). There are many formal definitions of a topology; we choose to work with the neighborhood space axiomatization [3] because it is most analogous to the data access model of index (element) in subset (neighborhood) of all indices (base space). In the indexing cube in Figure 1, the brown surface and blue rectangle cube are both neighborhoods of the index for the measurement in Albany on June 09. The blue rectangle is also a neighborhood of the index for the measurement in Albany on March 09. The indexing cube is a neighborhood for both of these indices. Using Munzner’s key-value based semantic indexing [17], the data in Figure 1 could be subset into sets of timeseries where the key would be station, or subset into maps where the key would be date, or subset into station records where the keys are (date, latitude, longitude). We encapsulate the desired properties of the connectivity as the category [12] $\mathcal{K}$ where the objects are all the neighborhoods and the morphisms are inclusion maps between neighborhoods [19]; these are the properties that are required as part of the implementation of the base space data type. Index spaces can be composed by taking the fibered coproduct, which is generalization of inserting non-duplicated records into a database.

Spivak shows that data typing can be expressed in a categorical framework in his fiber bundle formulation of tables in relational databases [20, 21]. We adopt Spivak’s definitions of *type*, *schema*, *record*, and *table* because that allows us to use a dimension agnostic named typing system for the dataset fields that is consistent with the abstraction we are using to express the continuity. As with the base space category $\mathcal{K}$, we propose a fiber category $\mathcal{F}$ with a single object $\mathbf{F}$ of an arbitrary data type and morphisms on the fiber object $\tilde{\phi} \in \text{Hom}(\mathbf{F}, \mathbf{F})^2$. Fields can be composed by taking the fiber product, which is a generalization of a database join on a key.

3.1.1 Example: Uniform Abstract Graphic Representation

We denote the output as a graphic, but the use of bundles allows us to generalize to output on any display space, such as a screen or 3D print. The graphic bundle consists of the total space $\mathbf{H}$ which

² $\text{Hom}(\text{source}, \text{target})$ denotes the set of maps from source to target

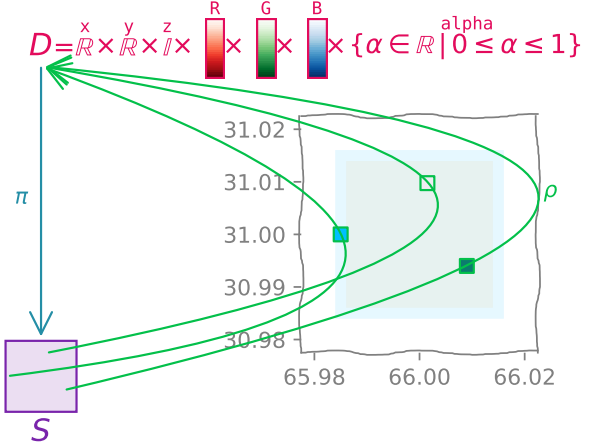


Figure 2: The pixels in the scatter marker are the records returned by the section function ρ over the points in the indexing space $s \in S$.

is an abstraction of an ideal (infinite resolution) space into which the graphic can be rendered, the base space $\mathbf{S}$ which is a parameterization of the display area (e.g. an inked bounding box), and the fiber space $\mathbf{D}$ which is an abstraction of the renderer fields (e.g. pixels). In Figure 2, the section function ρ maps into the fiber for a simplified 2D RGB infinite resolution pre-render space and returns the values of a pixel, approximated as the small colored boxes, in an infinite resolution space.

3.1.2 Data transformations

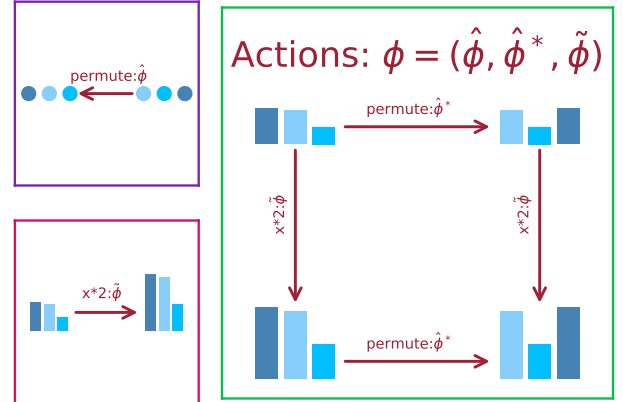


Figure 3: The actions on the base and fiber space commute such that the data can be permuted then scaled or scaled then permuted.

We separate data transformations into independent transformations on the base space ($\hat{\phi}, \hat{\phi}^*$) and on the fiber space $\tilde{\phi}$ and define a full data transformation as one that induces both a remapping of the index space and a change in the data values: $\phi : \tau \mapsto \hat{\phi} \circ \tau \circ \hat{\phi}^*$. We can define a composite transform where each fiber transform in a collection can be applied to a different fiber field, implemented in the same manner as record compositions. In Figure 3, the locations of the bars relative to each other change because the actions $\hat{\phi}, \hat{\phi}^*$ permute the indexing space. The action $\tilde{\phi}$ doubles the data value associated with the height of the bar. Because these are both structure preserving actions, the order - permute then scale or scale then

permute - does not change the final image.

3.2 Abstract Data Containers

While bundles provide a way to describe the structure of the data, sheaves are a mathematical way of describing the data container. Ghrist describes sheaves as an algebraic data structure; they provide a method for codifying the bookkeeping that data containers must implement to keep track of the continuity of the data [7]. Presheaves require that every change in index function $\hat{\phi}$ has a corresponding function $\hat{\phi}^*$ that aligns the records by "pulling back" records to where they belong in the new index. Sheaves are presheaves that satisfy the gluing and locality axioms: the gluing axiom says that the sections of the distributed representation of a dataset can be glued together and the locality axiom asserts that the glued representation is equivalent to a non-distributed representation over the same indexing space. When a data container implements sheaf constraints, the subsets and whole data space are guaranteed to have the same mathematical properties, e.g. morphisms such that visual algorithms written in our framework treat data that is in memory, distributed, streaming, or on-demand as equivalent.

4 CONNECTIVITY PRESERVING MAP: READABILITY ξ

From the early maps of stars on, there has been an expectation that the connectivity in visualizations matches the connectivity of the underlying data [6]. This is somewhat codified in the notion of readability [25], which is that visual elements must correspond to distinct data elements. We propose that if connectivity is preserved then there should exist a continuous map ξ that expresses the correspondence between graphic and data indexing spaces. We construct the map as going from graphic to data because that encodes the notion that every visual element traces back to the data in some way, but there may be more data available than what is shown in the graphic. As illustrated in Figure 4, modeling the data and graphic containers as sheaves and constructing structure preserving maps between the indices provides a method of precisely describing which data values belong with which graphical representations. In Figure 4, one example is that the orange bands on S_0, S_1, S_2 correspond to the orange point k_i and this correspondence allows us to construct a graphic specification for each data index $\xi_{*}\rho$. In Figure 4 $\xi_{*}\rho(K_i)$ a svg-like spec describing the piece of the circle, sine, and cosine curves generated for the data at the point k_i . We can also retrieve the data $\xi^*\tau_{\xi^{-1}(k)}$ that is being visualized by each specific part of the graphic, which in Figure 4 is the section $\tau(k_i)$.

5 ARTIST: STRUCTURE PRESERVING MAPS

We build off of Kindlmann and Scheidegger's notion of equivariant [10] and Mackinlay's notion of homomorphic [15] visualizations in our definition of structure preservation. We define a mapping from data to graphic to be **structure preserving** when connectivity is preserved, the structure on the data and the graphics is compatible, and changes to data and graphic commute. We call the structure preserving mappings **Artists**³. The output of an artist A is a restricted subset of graphic sections $\text{Im}_A(S, H)$ that are, by definition, only reachable through a structure preserving artist. The artist signature can be formulated as **Artist** : (**dataset** : **connectivity** $\rightarrow$ **fields**) $\rightarrow$ (**graphic** : **connectivity** $\rightarrow$ **fields**) where **connectivity** and **field** are meta classes of compatible types. A structurally preserving Artist is a *natural transform* [16, 12], which means that the Artist must preserve the constraints on the input sheaves and output sections, meaning that actions ϕ on data and graphic must be equivariant,

and the Artist must respect the sheaf constraints of *locality* and *gluing* for the input and equivalent output.

An artist is equivariant when changes to the input and output commute; for all points, starting at an arbitrary data point and transforming it into a different data point and then into a graphic is equivalent to transforming the original data point into a graphic and then transforming the graphic into another graphic. For example, Figure 5 shows an artist function A that maps a Dirichlet distribution: the artist is equivariant with respect to connectivity because the base space rotation actions $\hat{\phi}, \hat{\phi}^*$ commute such that rotating the data and then mapping it to a graphic or rotating the graphic yield the same Dirichlet distribution graphic where the values have moved to new locations but not changed and the artist is equivariant with respect to field structure because the binning action $\hat{\phi}$ commutes such that binning and then colormapping the data yields an equivalent graph to binning the colormap directly.

5.1 Composing Artists

Artists are composable because they preserve the properties of the coproduct and product [13] when composing with composable data; therefore, graphic composition is implemented correctly when artists return the same graphic on points in the intersection K^c and any shared fiber F^c is encoded consistently. This expectation of using the same encoding for the same variable is a generalization of the concept of consistency checking of multiple view encodings discussed by Qu and Hullman [18]. In Figure 6, this is a check that, e.g., the scatter marker drawn by A_a is identical to the scatter marker drawn by A_{abcd} . As illustrated in Figure 6, preserving these properties on compositions means that the same graphic can be generated from multiple artists drawing individual elements or a singular artist drawing the composite image. Structure preserving animations and interactions fall out as a consequence of implementing structure preserving artists because animation and interaction are simply a composition of base spaces (more frames) and/or fiber spaces (additions to the graphic such as annotations, or cursors).

6 CONSTRUCTING STRUCTURE PRESERVING COMPONENTS

Artists naturally factorize into a two-stage process, starting with an encoding stage v followed by a compositing stage Q . We encapsulate the space of components reachable through the encoding stage v as a visual fiber bundle V with sections μ . The restricted fiber space P of the bundle acts as the specification of the internal library representation of the visual components. We propose that reusing the fibers P across components facilitates standardizing internal types across the library and that this standardization improves maintainability.

In the **encoding** stage v , a data bundle is treated as separable fields where each field is mapped to a visual variable. In the **encoding** stage, many of the expected visual mappings η can be implemented inside the library. Factoring out the encoding stage leaves the **compositing** stage Q responsible for faithfully translating those visual components into a visual element. The encoder is infinitely composable $v' = v \circ v$ and the mark is composable with the encoder $Q' = Q \circ v$. The factorization of the artist into the encoding and composition stages provides a framework for testing the conversion from data to visual component separately from the assembly of components into a glyph. As shown in Figure 7, a set of v functions individually convert the values in the data record to visual components. Then the Q function combines these visual encodings to produce a graphic section ρ . When this section is evaluated on the graphic space associated with the data $\rho(\xi^{-1}(k))$, it produces green pixels at the intersection of the x and y positions listed in μ .

³ Artists are named for the Artist class in Matplotlib [9] because the Artist is the parent class of all the objects that assemble and manage visual elements such as points, lines, and shapes.

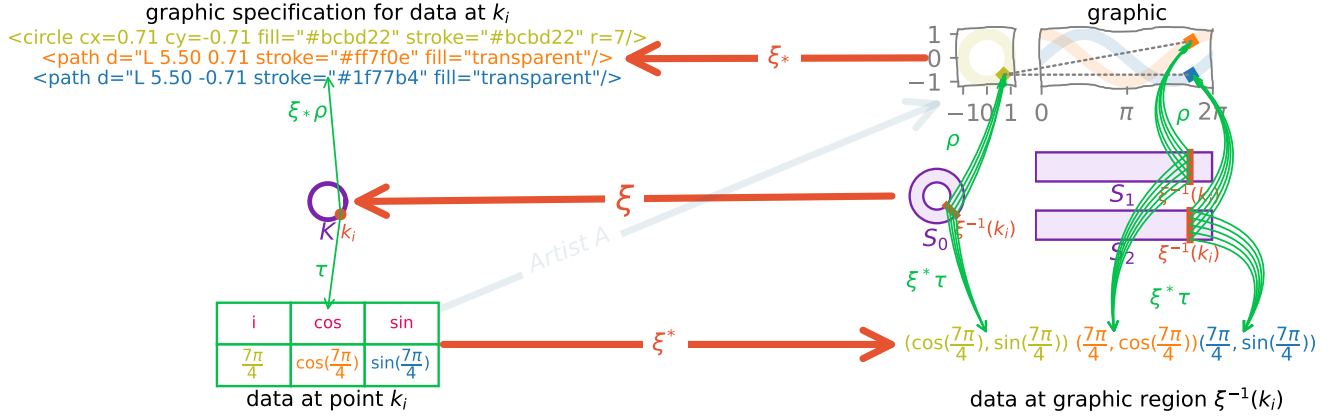


Figure 4: The data τ consists of the $\sin$ and $\cos$ functions over a unit circle base space K visualized as the circle and two line plots generated from the graphic specification ρ . The index map ξ keeps track of which part of the circle, sine, and cosine plots correspond to which point on the unit circle. The pushforward ξ_* matches each point in the data space to the corresponding graphic specification, and the pullback ξ^* matches each point in the graphic space to the data at that point.

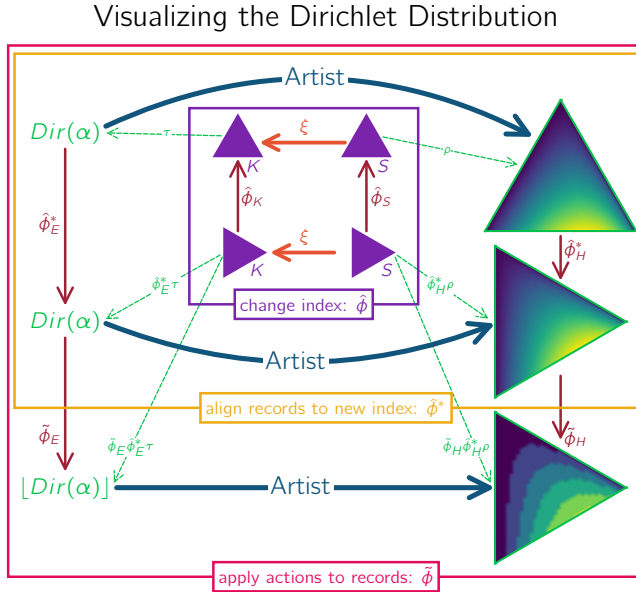


Figure 5: This artist maps a numerical section of the Dirichlet distribution to a graphical representation. The artist preserves connectivity by not breaking presheaf implementations of rotation ($\hat{\phi}, \hat{\phi}^*$) of the indexing space K, S and field structure by implementing the data to color mapping such that binning ($\hat{\phi}$) the data and then colormapping the bins is equivalent to binning the colors directly. The subscripts denote the space the action occurs in.

7 CONCLUSION

This paper demonstrates a mathematical formalism for representing the architecture of components in a data visualization library, in which the mathematical structures both enforce well known principles of data visualization and suggest factorizations and decompositions that result in highly modular infrastructure, such that artists can be constructed from composable primitives. There has been ongoing work to implement a new data container [11] in Matplotlib using ideas from our framework, which is adapted from

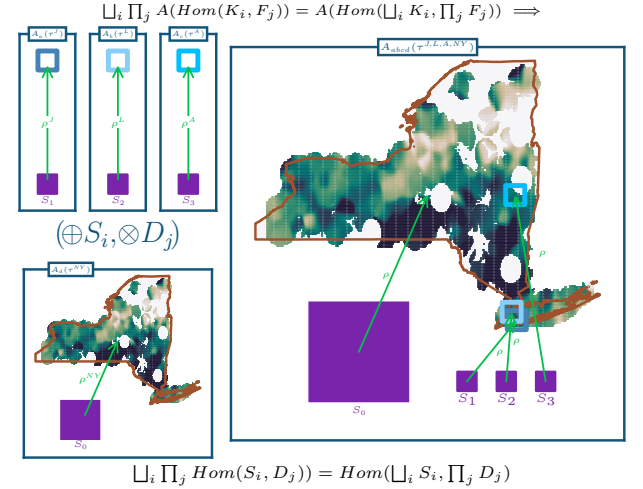


Figure 6: The scatter plot of stations on the rain map from Figure 1 can either be generated as a composition of scatter and map drawing functions or a singular scatter and map drawing function.

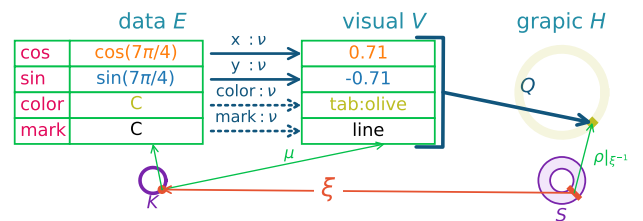


Figure 7: The encoder v maps the data values to visual values and the compositor Q generates the graphic specification for producing the sections of the circle. Default encodings, such as color and mark type, are here treated as pullbacks of constant values in the data.

<https://github.com/matplotlib/data-prototype>.

SUPPLEMENTAL MATERIALS

This paper summarizes a yet unpublished expanded manuscript at <https://github.com/story645/team/paper/>.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers who gave constructive feedback on an earlier version of this paper. The authors are also grateful to the various Matplotlib and Napari contributors, particularly Juan Nunez-Iglesias, and Nicolas Kruchten for their valuable feedback. Hannah is also very grateful to Nicolas for the suggestion of augmented notation and to the nlab and wikipedia contributors who wrote clear explanations of many of the topics discussed in this paper. This project has been made possible in part by grant number 2019-207333 (EOSS Cycle 1) and EOSS Cycle 3 from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation

REFERENCES

- [1] Adam Bartonicek, Simon Urbanek, and Paul Murrell. No More, No Less than Sum of Its Parts: Groups, Monoids, and the Algebra of Graphics, Statistics, and Interaction. 34(3):1063–1074.
- [2] Jacques Bertin. *Semiology of Graphics : Diagrams, Networks, Maps*. ESRI Press.
- [3] Ronald Brown. *Topology and Groupoids*.
- [4] D. M. Butler and M. H. Pendley. A visualization model based on the mathematics of fiber bundles. 3(5):45.
- [5] David M. Butler and Steve Bryson. Vector-Bundle Classes form Powerful Tool for Scientific Visualization. 6(6):576.
- [6] Michael Friendly. A Brief History of Data Visualization. In *Handbook of Data Visualization*, pages 15–56. Springer Berlin Heidelberg.
- [7] Robert W. Ghrist. *Elementary Applied Topology*, volume 1. Createspace Seattle.
- [8] Andrew Head, Amber Xie, and Marti A. Hearst. Math augmentation: How authors enhance the readability of formulas using novel visual design practices. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22. Association for Computing Machinery.
- [9] John Hunter and Michael Droettboom. The Architecture of Open Source Applications (Volume 2): Matplotlib.
- [10] G. Kindlmann and C. Scheidegger. An Algebraic Process for Visualization Design. 20(12):2181–2190.
- [11] Kyle Sunden. Proof of Concept: Data Containers by ksunden · Pull Request #30865 · matplotlib/matplotlib.
- [12] Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5. Springer Science & Business Media.
- [13] Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer.
- [14] J D Mackinlay. Automatic design of graphical presentations.
- [15] Jock Mackinlay. Automating the design of graphical presentations of relational information. 5(2):110–141.
- [16] Bartosz Milewski. *Category Theory for Programmers*. Bartosz Milewski.
- [17] Tamara Munzner. What: Data Abstraction. In *Visualization Analysis and Design*, AK Peters Visualization Series, pages 20–40. A K Peters/CRC Press.
- [18] Z. Qu and J. Hullman. Keeping multiple views consistent: Constraints, validations, and exceptions in visualization authoring. 24(1):468–477.
- [19] Emily Riehl. *Category Theory in Context*. Dover Publications.
- [20] David I Spivak. Databases are categories.
- [21] David I. Spivak. Simplicial databases.
- [22] S. S. Stevens. On the Theory of Scales of Measurement. 103(2684):677–680.
- [23] P. Vickers, J. Faith, and N. Rossiter. Understanding Visualization: A Formal Approach Using Category Theory and Semiotics. 19(6):1048–1061.
- [24] Leland Wilkinson. *The Grammar of Graphics*. Statistics and Computing. Springer-Verlag New York, Inc., 2nd ed edition.
- [25] Caroline Ziemkiewicz and Robert Kosara. Embedding Information Visualization within Visual Representation. In Zbigniew W. Ras and William Ribarsky, editors, *Advances in Information and Intelligent Systems*, pages 307–326. Springer Berlin Heidelberg.