TOPOLOGICAL EQUIVARIANT ARTIST MODEL

Abstract. The contract data visualization tools make with their users is that a chart is a faithful and accurate visual representation of the numbers it is made from. Motivated by wanting to make better tools, we propose a methodology for fully specifying arbitrary data to visualization mappings in a manner that easily translates to code. We propose that fiber bundles provide a uniform interface for describing a variety of underlying data - tables, images, networks, etc. - in a manner that independently encodes the mathematical structure of the topology and the fields of the dataset. Modeling the data structures that store the datasets as sheaves provide a method for specifying visualization methods that are designed to work regardless of how the dataset is stored - whether the data is 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 topology and fields of the data. Using category theory to formally express how visual elements are constructed means we can translate those expectations into code, which can then be used to enforce the expectation that a visualization tool is faithfully translating between numbers and charts.

1. Intro

Motivated by wanting to make better tools, we propose a methodology for fully specifying arbitrary data to visualization mappings in a manner that easily translates to code. Generally, preserving structure means that a visualization is expected to preserve the *field* properties and *topology* of the corresponding dataset, where **field** is a set of values of the same type and **topology** is the connectivity and relative positioning of elements in a dataset [13]. Field structure is traditionally codified as the Steven's measurement scales [11], where each scale is a set of actions on a group Topological structure is generally assumed by visual algorithms [4,12], but generally do not verify that input structure. For example, a `line` algorithm often does not have a way to query whether a list of (x,y) coordinates is the distinct rows, the time series, or the list of stations in Figure 1. We propose that the bar plot, line plot, and heatmap can be verified as structure preserving because they have a homeomorphic relationship to the 0D (●) points, 1D (–) linear, and 2D (■) surface continuities embedded in the continuous 3 dimensional. Our methodology generalizes Bertin [1]'s codification of structure, Mackinlay [8] mathematical formalization, Wilkinson's set theoretic approach [13], and Kindlemann and Scheidegger's algebraic visualization design (AVD) framework [7] by explicitly incorporating topology, supporting non-group and non-monoidal structures, and by providing a framework for translating the theoretical ideas into buildable components.

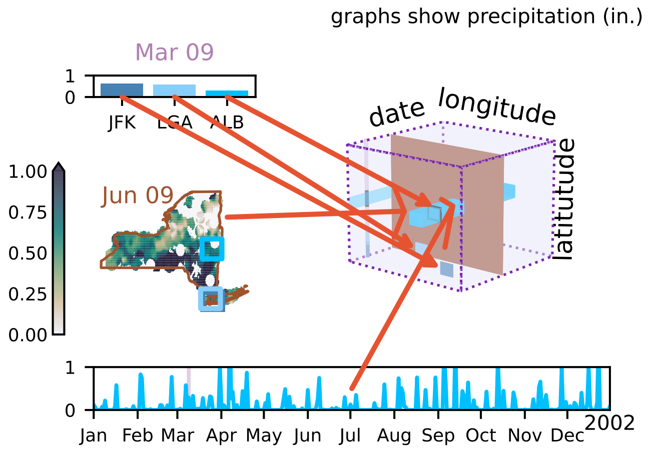


Figure 1. This weather station data has multiple embedded continuities - points at each time and position, timeseries at each position, and maps at each time - that are expected to be homeomorphic to the continuity of their respective visualizations.

2. Data Model

We extend Butler's proposal of using fiber bundles as an abstraction for visualization data [2,3] by incorporating Spivak's methodology for encoding named data types from his fiber bundle representation of relational databases [9,10].

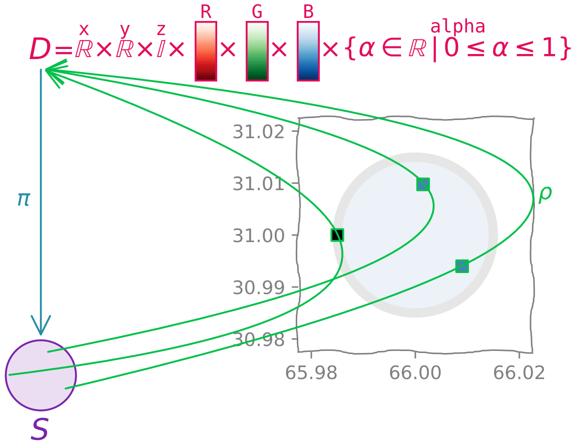


Figure 2. The scatter marker is specified by the section ρ, which maps into the fiber D to retrieve the values that compose into the pixel (approximated as a square) returned by the section function evaluated at each point s. The section evaluated over the entire space $\rho|_S$ returns the entire scatter mark, shown here in faded form to make it easier to see the individual pixels.

structure/abstraction	Data	Visual	Graphic
fiber/total /base spaces fields/datasets/topology	$F \hookrightarrow E \xrightarrow{\pi} K$	$P \hookrightarrow V \xrightarrow{\pi} K$	$D \hookrightarrow H \xrightarrow{\pi} S$
point/openset/base space index/subset/indices	$k \in U \subseteq K$	$k \in U \subseteq K$	$s \in W \subseteq S$
fiber space record/fields	$r \in F$	$V \in P$	$d \in D$
section dataset	$\Gamma(K, E) \ni \tau : K \to F$	$\Gamma(K, V) \ni \mu : K \to P$	$\Gamma(S, H) \ni \rho : S \to D$
sheaf data container	$\mathcal{O}_{K,E} : U \to \Gamma(U, E \restriction_U)$	$\mathcal{O}_{K,V} : U \to \Gamma(U, V \restriction_U)$	$\mathcal{O}_{S,H} : W \to \Gamma(U, H \restriction_W)$

TABLE 1. Summary of how the data is abstracted using topological structures

The indexing space is modeled as a topology $\mathcal{T}$ on the underlying data indexing space because it is a uniform method of describing arbitrary continuity and it provides a method for managing overlapping embedded topologies, such as Figure 1. Adopting Spivak's fiber bundle construction of types allows our model to reuse types so long as the field names are distinct and encode multidimensional fields can be encoded in the same manner as single dimensional fields because domains of types are a mathematical space. We encode data as a section τ of a bundle because this allows us to incorporate the topology and field types in the data definition `dataset` : `topology` → `field`. Abstracting the data containers as sheaves provides a way of encoding how data containers keep track of the continuity of the data [5] such that visual algorithms developed on this model apply whether the data fits in memory, is distributed, or is streaming.

3. ARTIST

We call the mapping from data to graphic $A : \tau \mapsto \rho$ an Artist[1] and propose that an artist is structure preserving when the artist is implemented as a natural transform on sheaves. This means that the artist satisfies the following constraints, as illustrated in Figure 3:

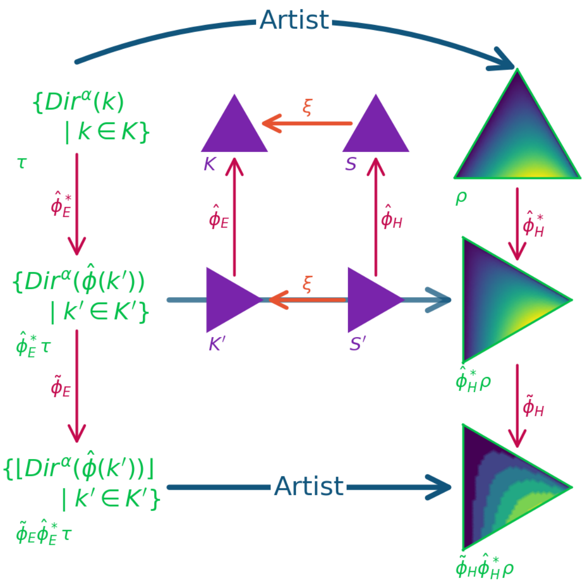


FIGURE 3

homeomorphic: there exists a deformation retract $\xi : S \to K$ between graphic and artist base space

topology: the artist is commutative w.r.t. the presheaf morphism $\hat{\phi} : k' \mapsto k$ and pullback $\hat{\phi}^*\tau \restriction_{U'} : \tau \restriction_{U'} \mapsto \tau \restriction_{U' \circ \hat{\phi}}$ such that when an index of the record changes the values in the record are unmodified $\tau(K) = \hat{\phi}^*\tau(k') = \tau(\hat{\phi}(k'))$

fields: the artist is commutative w.r.t. actions on the fiber $\tilde{\phi} \in Hom(F, F)$, which are the functions between sections $\tilde{\phi} : \hat{\phi}^*\tau \restriction_{U'} \mapsto \hat{\phi}^*\tau' \restriction_U$. Additionally, $\tilde{\phi}$ must preserve features of F, such as operators, that are defined as part of the structure of F.

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

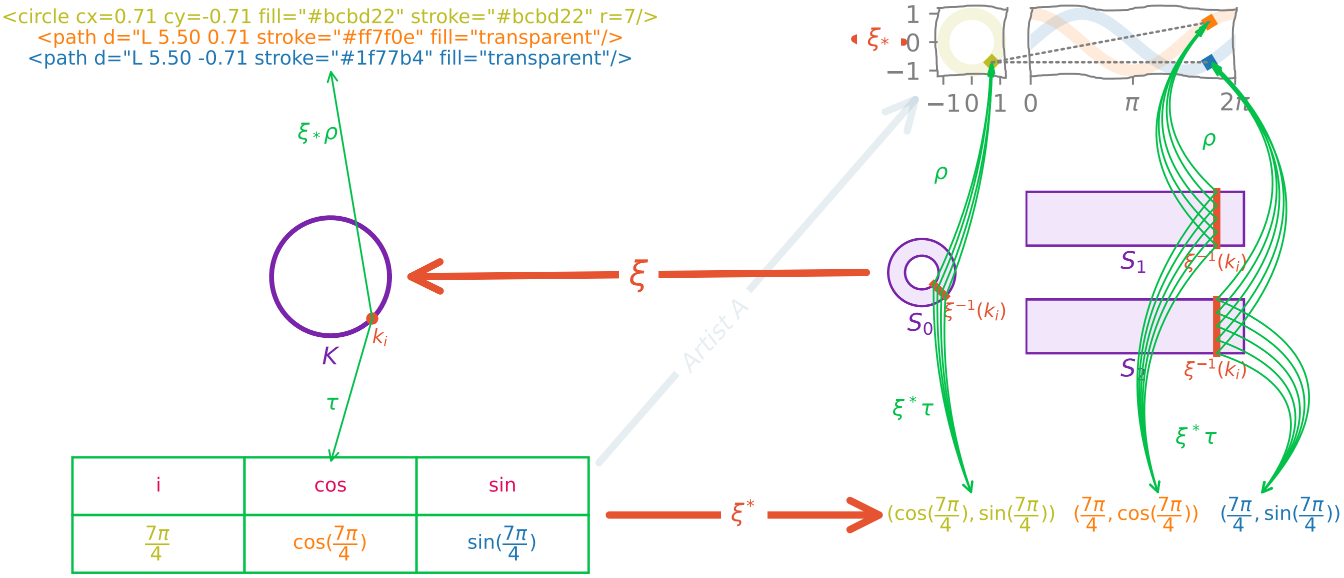
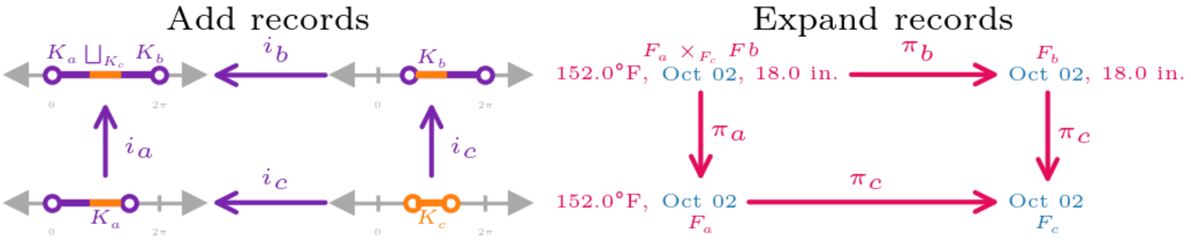

<circle cx=0.71 cy=-0.71 fill="#bcbd22" stroke="#bcbd22" r=7/>
<path d="L 5.50 0.71 stroke="#ff7f0e" fill="transparent"/>
<path d="L 5.50 -0.71 stroke="#1f77b4" fill="transparent"/>
$\xi_*\rho$
k_i
K
τ
ξ
Artist A
ξ_*
1
0
−1
−1 0 1
0
π
2π
ρ
ρ
S_0
$\xi^{-1}(k_i)$
S_1
$\xi^{-1}(k_i)$
S_2
$\xi^{-1}(k_i)$
$\xi^*\tau$
$\xi^*\tau$
i
cos
sin
$\frac{7\pi}{4}$
$\cos(\frac{7\pi}{4})$
$\sin(\frac{7\pi}{4})$
ξ^*
$(\cos(\frac{7\pi}{4}), \sin(\frac{7\pi}{4}))$
$(\frac{7\pi}{4}, \cos(\frac{7\pi}{4}))$
$(\frac{7\pi}{4}, \sin(\frac{7\pi}{4}))$

FIGURE 4. The graphics over the orange bands map to the data at the orange point.

As illustrated in 4, the transport functors (ξ_*, ξ^*) constructed from ξ [6] also codify how visual elements correspond to distinct data elements, which is a necessary condition of a visualization being readable [14]. The pushforward ξ_* matches each point in the data space to the specification of the graphic at that point $\xi_*\rho(k) = \rho\restriction_{\xi^{-1}(k)}$, while the pullback ξ^* matches each point in the graphic space to the data over that point $\xi^*\tau(s) = \tau(\xi(s)) = \tau(k)$.



The topology and fiber categories are equipped with operators to add and extend records, respectively, Since the artist preserves these operators, artists have the property of being composable $A(\bigsqcup_i K_i, \prod_j F_j) = \bigsqcup_i \prod_j A(K_i, F_j)$

4. CONSTRUCTION

The artist can be constructed with an encoding stage ν and a compositing stage Q s.t $A = Q \circ \nu$. In the encoding stage ν, a data bundle is mapped to a visual variable bundle ν, where the fibers of ν are an internal library specification. The compositing stage Q is then responsible for assembling the visual components into a visual element.

Note

This is a work in progress, more information can be found at https://github.com/story645/team

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 from the library developer perspective. 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 Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation

References

[1] Jacques Bertin. *Semiology of Graphics : Diagrams, Networks, Maps.* ESRI Press.

[2] D. M. Butler and M. H. Pendley. A visualization model based on the mathematics of fiber bundles. 3(5):45.

[3] David M. Butler and Steve Bryson. Vector-Bundle Classes form Powerful Tool for Scientific Visualization. 6(6):576.

[4] E. H. Chi. A taxonomy of visualization techniques using the data state reference model. In *IEEE Symposium on Information Visualization 2000. INFOVIS 2000. Proceedings*, pages 69–75.

[5] Robert W. Ghrist. *Elementary Applied Topology*, volume 1. Createspace Seattle.

[6] G. Harder and K. Diederich. *Lectures on Algebraic Geometry I: Sheaves, Cohomology of Sheaves, and Applications to Riemann Surfaces.* Aspects of Mathematics. Vieweg+Teubner Verlag.

[7] G. Kindlmann and C. Scheidegger. An Algebraic Process for Visualization Design. 20(12):2181–2190.

[8] Jock Mackinlay. Automatic Design of Graphical Presentations.

[9] David I Spivak. Databases are categories.

[10] David I. Spivak. Simplicial databases.

[11] S. S. Stevens. On the Theory of Scales of Measurement. 103(2684):677–680.

[12] M. Tory and T. Moller. Rethinking visualization: A high-level taxonomy. In *IEEE Symposium on Information Visualization*, pages 151–158.

[13] Leland Wilkinson. *The Grammar of Graphics.* Statistics and Computing. Springer-Verlag New York, Inc., 2nd ed edition.

[14] 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.