1 Topological Artist Model

As discussed in the introduction, visualization is generally defined as structure preserving maps from a data object to a graphic object. In order to formalize this statement, we describe the connectivity of the records using topology and define the structure on the components in terms of the monoid actions on the component types. By formalizing structure in this way, we can evaluate the extent to which a visualization preserves the structure of the data it is representing and build structure preserving visualization tools. We introduce the notion of an artist $\mathscr{A}$ as an equivariant map from data to graphic

$$\mathscr{A} : \mathscr{E} \to \mathscr{H} \tag{1}$$

that carries a homomorphism of monoid actions $\varphi : M \to M'$ [1], which are discussed in detail in section 1.1.2. Given M on data $\mathscr{E}$ and M' on graphic $\mathscr{H}$, we propose that artists $\mathscr{A}$ are equivariant maps

$$\mathscr{A}(m \cdot r) = \varphi(m) \cdot \mathscr{A}(r) \tag{2}$$

such that applying a monoid action $m \in M$ to the data $r \in \mathscr{E}$ input to $\mathscr{A}$ is equivalent to applying a monoid action $\varphi(M) \in M'$ to the graphic $A(r) \in \mathscr{H}$ output of the artist.

We model the data $\mathscr{E}$, graphic $\mathscr{H}$, and intermediate visual encoding $\mathscr{V}$ stages of visualization as topological structures that encapsulate types of variables and continuity. To explain which structure the artist is preserving, we first describe how we model data (1.1), graphics (1.2), and intermediate visual characteristics (1.3) as fiber bundles. We then discuss the equivariant maps between data and visual characteristics (1.3.2) and visual characteristics and graphics (1.3.3) that make up the artist.

1.1 Data Space E

Building on Butler's proposal of using fiber bundles as a common data representation structure for visualization data[2, 3], a fiber bundle is a tuple (E, K, π, F) defined by the projection map π

$$F \hookrightarrow E \xrightarrow{\pi} K \tag{3}$$

that binds the components of the data in F to the continuity represented in K. The fiber bundle models the properties of data component types F (1.1.1), the continuity of records K (1.1.3), the collections of records τ (1.1.4), and the space E of all possible datasets with these components and continuity.

By definition fiber bundles are locally trivial[4, 5], meaning that over a localized neighborhood we can dispense with extra structure on E and focus on the components and continuity. We use fiber bundles as the data model because they are inclusive enough to express all the types of data described in section **??**.

1.1.1 Variables in Fiber Space F

To formalize the structure of the data components, we use notation introduced by Spivak [6] that binds the components of the fiber to variable names. This allows us to describe the components in a schema like way. Spivak constructs a set $\mathbb{U}$ that is the disjoint union of all possible objects of types $\{T_0, \ldots, T_m\} \in \mathbf{DT}$, where $\mathbf{DT}$ are the data types of the variables in the dataset. He then defines the single variable set $\mathbb{U}_\sigma$

$$\begin{tikzcd}
\mathbb{U}_\sigma \arrow[r] \arrow[d, "\pi_\sigma"'] & \mathbb{U} \arrow[d, "\pi"] \\
C \arrow[r, "\sigma"'] & \mathbf{DT}
\end{tikzcd} \tag{4}$$

which is $\mathbb{U}$ restricted to objects of type T bound to variable name c. The $\mathbb{U}_\sigma$ lookup is by name to specify that every component is distinct, since multiple components can have the same type T. Given σ, the fiber for a one variable dataset is

$$F = \mathbb{U}_{\sigma(c)} = \mathbb{U}_T \tag{5}$$

where σ is the schema binding variable name c to its datatype T. A dataset with multiple variables has a fiber that is the cartesian cross product of $\mathbb{U}_\sigma$ applied to all the columns:

$$F = \mathbb{U}_{\sigma(c_1)} \times \ldots \mathbb{U}_{\sigma(c_i)} \ldots \times \mathbb{U}_{\sigma(c_n)} \tag{6}$$

which is equivalent to

$$F = F_0 \times \ldots \times F_i \times \ldots \times F_n \tag{7}$$

which allows us to decouple F into components F_i.

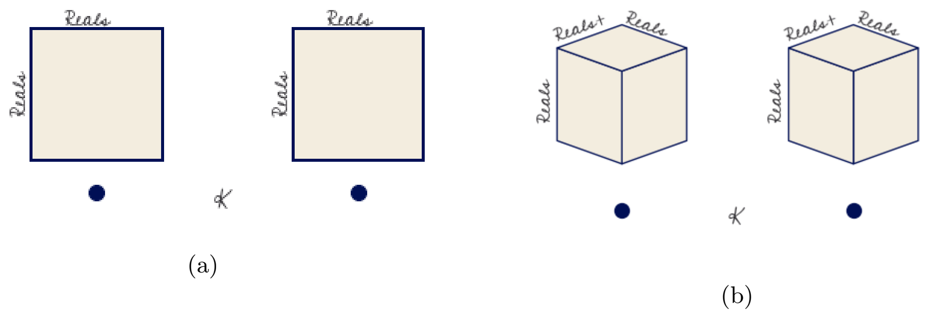

Figure 1: These two datasets have the same base space K of discrete points, but figure 1a has fiber $F = \mathbb{R} \times \mathbb{R}$ which is (time, temperature) while figure 1b has fiber $\mathbb{R} \times \mathbb{R}^+ \times \mathbb{R}$ which is (time, wind=(speed, direction))

For example, the data in figure 1a is a pair of times and °K temperature measurements taken at those times. Time is a positive number of type `datetime` which can be resolved to floats $\mathbb{U}_{\texttt{datetime}} = \mathbb{R}$. Temperature values are real positive numbers $\mathbb{U}_{\texttt{float}} = \mathbb{R}^+$. The fiber is

$$\mathbb{U} = \mathbb{R} \times \mathbb{R}^+ \tag{8}$$

where the first component F_0 is the set of values specified by ($c = \textit{time}$, $T = \texttt{datetime}$, $\mathbb{U}_\sigma = \mathbb{R}$) and F_1 is specified by ($c = \textit{temperature}$, $T = \texttt{float}$, $\mathbb{U}_\sigma = \mathbb{R}^+$) and is the set of values $\mathbb{U}_\sigma = \mathbb{R}^+$. In figure 1b, temperature is replaced with wind. This wind variable is of type wind and has two components speed and direction $\{(s, d) \in \mathbb{R}^2 \mid 0 \leq s, 0 \leq d \leq 360\}$. Therefore, the fiber is

$$F = \mathbb{R}^+ \times \mathbb{R}^2 \tag{9}$$

such that F_1 is specified by ($c = \mathit{wind}$, $T =$ `wind`, $\mathbb{U}_\sigma = \mathbb{R}^2$). As illustrated in figure 1, Spivak's framework provides a consistent way to describe potentially complex components of the input data.

1.1.2 Measurement Scales: Monoid Actions

Implementing expressive visual encodings requires formally describing the structure on the components of the fiber, which we define by the actions of a monoid on the component. In doing so, we specify the properties of the component that must be preserved in a graphic representation. While structure on a set of values is often described algebraically as operations or through the actions of a group, for example Steven's scales [7], we generalize to monoids to support more component types. Monoids are also commonly found in functional programming because they specify compositions of transformations [8, 9].

A monoid [10] M is a set with an associative binary operator $* : M \times M \to M$. A monoid has an identity element $e \in M$ such that $e * a = a * e = a$ for all $a \in M$. As defined on a component of F, a left monoid action [11, 12] of M_i is a set F_i with an action $\bullet : M \times F_i \to F_i$ with the properties:

associativity for all $f, g \in M_i$ and $x \in F_i$, $f \bullet (g \bullet x) = (f * g) \bullet x$
identity for all $x \in F_i, e \in M_i$, $e \bullet x = x$

As with the fiber F the total monoid space M is the cartesian product

$$M = M_0 \times \ldots \times M_i \times \ldots \times \ldots M_n \tag{10}$$

of each monoid M_i on F_i. The monoid is also added to the specification of the fiber (c_i, T_i, $\mathbb{U}_\sigma$ M_i)

Steven's described the measurement scales[7, 13] in terms of the monoid actions on the measurements: nominal data is permutable, ordinal data is monotonic, interval data is translatable, and ratio data is scalable [14]. For example, given an arbitrary interval scale fiber component ($c = \mathit{temperature}$, $T =$ `float`, $\mathbb{U}_\sigma = \mathbb{R}$) with with arbitrary monoid translation actions chosen for this example:

- monoid operator addition $* = +$
- monoid operations: $f : x \mapsto x + 1°C$, $g : x \mapsto x + 2°C$
- monoid action operator composition $\bullet = \circ$

By structure preservation, we mean that monoid actions are composable. For the translation actions described above on the temperature fiber, this means that they satisfy the condition

$$\begin{tikzcd}
\mathbb{R} \arrow[d, "x+1°"'] \arrow[dr, "(x+1°C)\circ(x+2°C)"] & \\
\mathbb{R} \arrow[r, "x+2°C"'] & \mathbb{R}
\end{tikzcd} \tag{11}$$

where $1°C$ and $2°C$ are valid distances between two temperatures x. What this diagram means is that either the fiber could be shifted by $1°C$ (vertical line) then by $2°C$ (horizontal), or the two shifts could be combined such that in this case the fiber is shifted by $3°C$ (diagonal) and these two paths yield the same temperature.

While many component types will be one of the measurement scale types, we generalize to monoids specifically for the case of partially ordered set. Given a set $W = \{mist, drizzle, rain\}$, then the map $f : W \to W$ defined by

1. $f(rain) = drizzle$,
2. $f(drizzle) = mist$
3. $f(mist) = mist$

is order preserving such that $mist \leq drizzle \leq rain$ but has no inverse since *drizzle* and *mist* go to the same value *mist*. Therefore order preserving maps do not form a group, and instead we generalize to monoids to support partial order component types. Defining the monoid actions on the components serves as the basis for identifying the invariance[**kindlmann2014algebraic**] that must be preserved in the visual representation of the component. We propose equivariance of monoid actions individually on the fiber to visual component maps and on the graphic as a whole.

1.1.3 Continuity of the Data K

The base space K is way to express how the records in E are connected to each other, for example if they are discrete points or if they lie in a 2D continous surface. Connectivity type is assumed in the choice of visualization, for example a line plot implies 1D continuous data, but an explicit representation allows for verifying that the topology of the graphic representation is equivalent to the topology of the data.

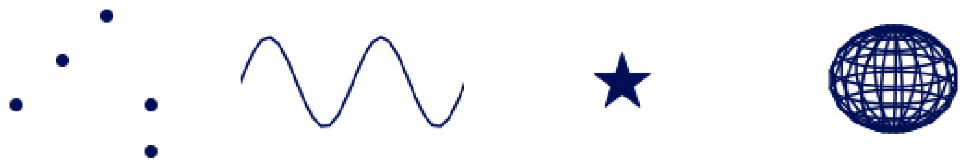

Figure 2: The topological base space K encodes the connectivity of the data space, for example if the data is independent points or on a plane or a sphere

As illustrated in figure 2, K is akin to an indexing space into E that describes the structure of E. K can have any number of dimensions and can be continuous or discrete.

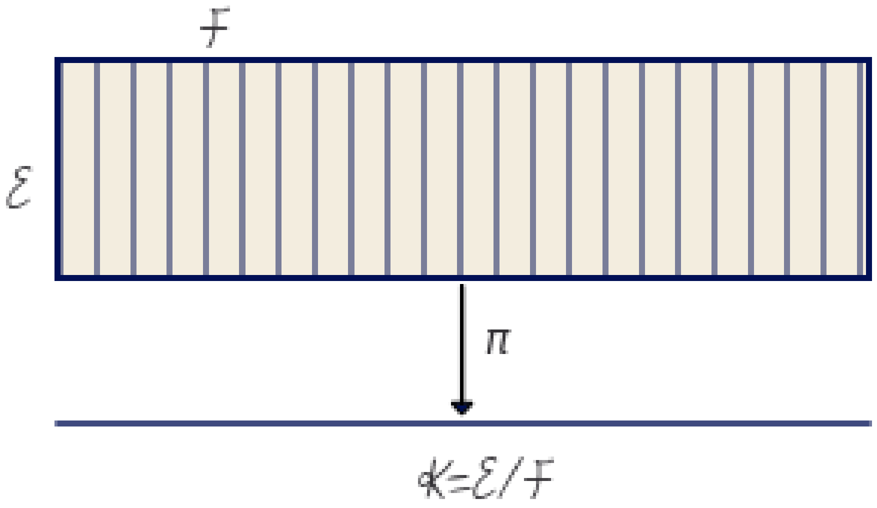

Figure 3: The base space E is divided into fiber segments F. The base space K acts as an index into the records in the fibers.

Formally K is the quotient space [15] of E meaning it is the finest space[16] such that every $k \in K$ has a corresponding fiber F_k[15]. In figure 3, E is a rectangle divided by vertical fibers F, so the minimal K for which there is always a mapping $\pi : E \to K$ is the closed interval $[0, 1]$. As with fibers and monoids, we can decompose the total space into components $\pi : E_i \to K$ where

$$\pi : E_1 \oplus \ldots \oplus E_i \oplus \ldots \oplus E_n \to K \tag{12}$$

which is a decomposition of F. The K remains the same because the connectivity of records does not change just because there are fewer elements in each record.

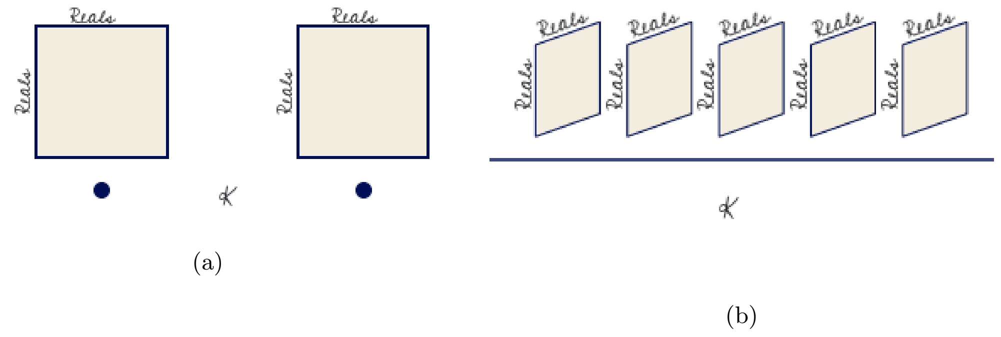

Figure 4: These two datasets have the same (time, temperature) fiber. In figure 4a the total space E is discrete over points $k \in K$, meaning the records in the fiber are also discrete. In figure 4b E lies over the continuous interval K, meaning the records in the fiber are sampled from a continuous space.

The datasets in figure 4 have the same fiber of (temperature, time). In figure 4a the fibers lie over discrete K such that the records in the datasets in the fiber bundles are discrete. The same fiber in figure 4b lies over a continuous interval K such that the records are samples from a continuous function defined on K. By encoding this continuity in the model as K the data model now explicitly carries information about its structure such that the implicit assumptions of the visualization algorithms are now explicit. The explicit topology is a concise way of distinguishing visualizations that appear identical, for example heatmaps and images.

1.1.4 Data τ

While the projection function $\pi : E \to K$ ties together the base space K with the fiber F, a section $\tau : K \to E$ encodes a dataset. A section function takes as input location $k \in K$ and returns a record $r \in E$. For example, in the special case of a table [6], K is a set of row ids, F is the columns, and the section τ returns the record r at a given key in K. For any fiber bundle, there exists a map

$$\begin{tikzcd} F \arrow[r, hook] & E \arrow[d, "\pi"', shift right] \\ & K \arrow[u, "\tau"', bend right, shift right] \end{tikzcd} \tag{13}$$

such that $\pi(\tau(k)) = k$. The set of all global sections is denoted as $\Gamma(E)$. Assuming a trivial fiber bundle $E = K \times F$, the section is

$$\tau(k) = (k, (g_{F_0}(k), \dots, g_{F_n}(k))) \tag{14}$$

where $g : K \to F$ is the index function into the fiber. This formulation of the section also holds on locally trivial sections of a non-trivial fiber bundle. Because we can decompose the bundle and the fiber, we can decompose τ as

$$\tau = (\tau_0, \dots, \tau_i, \dots, \tau_n) \tag{15}$$

where each section τ_i is a variable or set of variables. This allows for accessing the data component wise in addition to accessing the data in terms of its location over K.

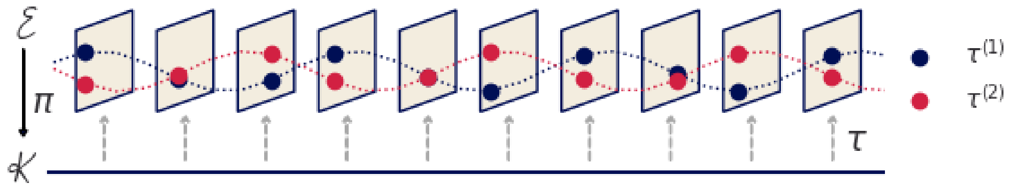

Figure 5: Fiber (time, temperature) with an interval K basespace. The sections $\tau^{(1)}$ and $\tau^{(2)}$ are constrained such that the time variable must be monotonic, which means each section is a timeseries of temperature values. They are included in the global set of sections $\tau^{(1)}, \tau^{(2)} \in \Gamma(E)$

In the example in figure 5, the fiber is (*time*, *temperature*) as described in figure 1 and the base space is the interval K. The section $\tau^{(1)}$ resolves to a series of monotonically increasing in time records of (time, temperature) values. Section $\tau^{(2)}$ returns a different timeseries of (time, temperature) values. Both sections are included in the global set of sections $\tau^{(1)}, \tau^{(2)} \in \Gamma(E)$.

1.1.5 Applications to Data Containers

This model provides a common formalism for widely used data containers without sacrificing the semantic structure embedded in each container. For example, the section can be any instance of a univariate numpy array[17] that stores an image. This could be a section of a fiber bundle where K is a 2D continuous plane and the F is $(\mathbb{R}^3, \mathbb{R}, \mathbb{R})$ where $\mathbb{R}^3$ is color, and the other two components are the x and y positions of the sampled data in the image. This position information is already implicitely encoded in the array as the index and the resolution of the image being stored.Instead of an image, the numpy array could also store a 2D discrete table. The fiber would not change, but the K would now be 0D discrete points. These different choices in topology indicate, for example, what sorts of interpolation would be appropriate when visualizing the data.

There are also many types of labeled containers that can richly be described in this framework because of the schema like structure of the fiber. For example, a pandas series which stores a labeled list, or a dataframe[18] which stores a relational table. A series could store the values of $\tau^{(1)}$ and a second series could be $\tau^{(2)}$. We could also fatten the fiber to hold two temperature series, such that a section would be an instance of a dataframe with a time column and two temperature columns. While the series and dataframe explicitly have a time index column, they are components in our model and the index is assumed to be data independent references such as hashvalues, virtual memory locations, or random number keys.

Where this model particularly shines are N dimensional labeled data structures. For example, an xarray[19] data that stores temperature field could have a K that is a continuous volume and the components would be the temperature and the time, latitude, and longitude the measurements were sampled at. A section can also be an instance of a distributed data container, such as a dask array [20]. As with the other containers, K and F are defined in terms of the index and dtypes of the components of the array. Because our framework is defined in terms of the fiber, continuity, and sections, rather than the exact values of the data, our model does not need to know what the exact values are until the renderer needs to fill in the image.

1.2 Graphic Space H

We introduce a graphic bundle to hold the essential information necessary to render a graphical design constructed by the artist. As with the data, we can represent the target graphic as a section ρ of a bundle (H, S, π, D). The graphic bundle H consists of a base S(1.2.1) that is a thickened form of K a fiber D(1.2.2) that is an idealized display space, and sections ρ(1.2.3) that encode a graphic where the visual characteristics are fully specified.

1.2.1 Idealized Display D

To fully specify the visual characteristics of the image, we construct a fiber D that is an infinite resolution version of the target space. Typically H is trivial and therefore sections can be thought of as mappings into D. In this work, we assume a 2D opaque image $D = \mathbb{R}^5$ with elements

$$(x,\, y,\, r,\, g,\, b) \in D \tag{16}$$

such that a rendered graphic only consists of 2D position and color. To support overplotting and transparency, the fiber could be $D = \mathbb{R}^7$ such that $(x, y, z, r, g, b, a) \in D$ specifies the target display. By abstracting the target display space as D, the model can support different targets, such as a 2D screen or 3D printer.

1.2.2 Continuity of the Graphic S

Just as the K encodes the connectivity of the records in the data, we propose an equivalent S that encodes the connectivity of the rendered elements of the graphic. For example, consider a S that is mapped to the region of a 2D display space that represents K. For some visualizations, K may be lower dimension than S. For example, a point that is 0D in K cannot be represented on screen unless it is thickened to 2D to encode the connectivity of the pixels that visually represent the point. This thickening is often not necessary when the dimensionality of K matches the dimensionality of the target space, for example if K is 2D and the display is a 2D screen. We introduce S to thicken K in a way which preserves the structure of K.

Formally, we require that K be a deformation retract[21] of S so that K and S have the same homotopy. The surjective map $\xi : S \to K$

$$\begin{tikzcd} E \arrow[d, "\pi"'] & H \arrow[d, "\pi"'] \\ K & S \arrow[l, "\xi"'] \end{tikzcd} \tag{17}$$

goes from region $s \in S_k$ to its associated point s. This means that if $\xi(s) = k$, the record at k is copied over the region s such that $\tau(k) = \xi^*\tau(s)$ where $\xi^*\tau(s)$ is τ pulled back over S.

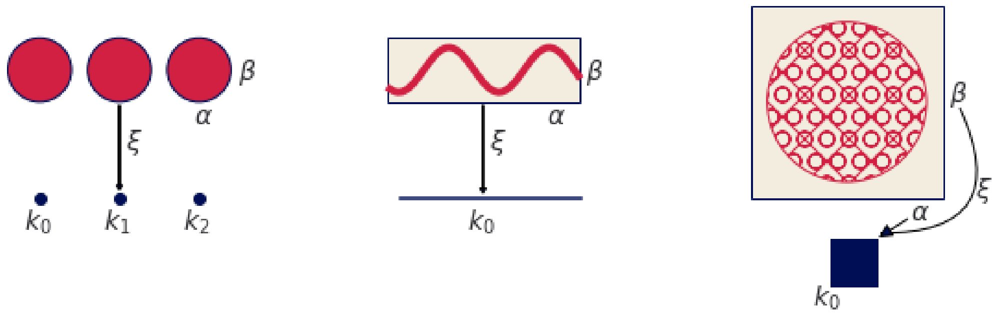

Figure 6: The scatter and line graphic base spaces have one more dimension of continuity than K so that S can encode physical aspects of the glyph, such as shape (a circle) or thickness. The image has the same dimension in S as in K.

When K is discrete points and the graphic is a scatter plot, each point $k \in K$ corresponds to a 2D disk S_k as shown in figure 6. In the case of 1D continuous data and a line plot, the region β over a point α_i specifies the thickness of the line in S for the corresponding τ on k. The image has the same dimensions in data space and graphic space such that no extra dimensions are needed in S.

The mapping function ξ provides a way to identify the part of the visual transformation that is specific to the the connectivity of the data rather than the values; for example it is common to flip a matrix when displaying an image. The ξ mapping is also used by interactive visualization components to look up the data associated with a region on screen. One example is to fill in details in a hover tooltip, another is to convert region selection (such as zooming) on S to a query on the data to access the corresponding record components on K.

1.2.3 Graphic ρ

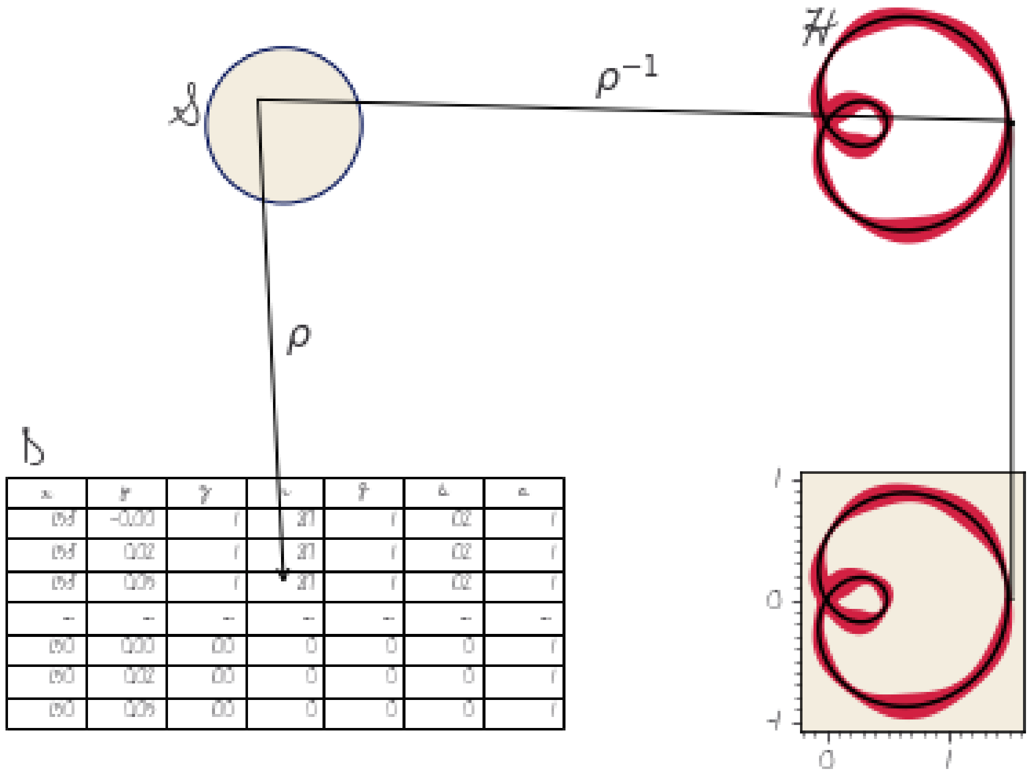

Figure 7: To render a graphic, a pixel p is selected in the display space, which is defined in the same coordinates as the x and y components in D. The inverse mapping $\rho_{xy}{}^{(}p)$ returns a region $S_p \subset S$. $\rho(S_p)$ returns the list of elements $(x, y, r, g, b) \in D$ that lie over S_p. The integral over the (r, g, b) elements is the color of the pixel.

This section describes how we go from a graphic in an idealized prerender space to a rendered image, where the graphic is the section $\rho : S \to H$. It is sufficient to sketch out how an arbitrary pixel would be rendered, where a pixel p in a real display corresponds to a region S_p in the idealized display. To determine the color of the pixel, we aggregate the color values over the region via integration.

For a 2D screen, the pixel is defined as a region $p = [y_{top}, y_{bottom}, x_{right}, x_{left}]$ of the rendered graphic. Since the x and y in p are in the same coordinate system as the x and y components of D the inverse map of the bounding box $S_p = {\rho_{xy}}^{-1}(p)$ is a region $S_p \subset S$. To compute the color, we integrate on S_p

$$r_p = \iint\limits_{S_p} \rho_r(s) ds^2 \tag{18}$$

$$g_p = \iint\limits_{S_p} \rho_g(s) ds^2 \tag{19}$$

$$b_p = \iint\limits_{S_p} \rho_b(s) ds^2 \tag{20}$$

As shown in figure 7, a pixel p in the output space is selected and inverse mapped into the corresponding region $S_p \subset S$. This triggers a lookup of the ρ over the region S_p, which yields the set of elements in D that specify the (r, g, b) values corresponding to the region p. The color of the pixel is then obtained by taking the integral of $\rho_{rgb}(S_p)$.

In general, ρ is an abstraction of rendering. In very broad strokes ρ can be a specification such as PDF[22], SVG[23], or an openGL scene graph[24]. Alternatively, ρcan be a rendering engine such as cairo[25] or AGG[26]. Implementation of ρ is out of scope for this work,

1.3 Artist

We propose that the transformation from data to visual representation can be described as a structure preserving map from one topological space to another. We name this map the artist as that is the analogous part of the Matplotlib[27] architecture that builds visual elements. The topological artist A is a monoid equivariant sheaf map from the sheaf on a data bundle E which is $\mathcal{O}(E)$ to the sheaf on the graphic bundle H, $\mathcal{O}(H)$.

$$A : \mathcal{O}(E) \to \mathcal{O}(H) \tag{21}$$

Sheafs are a mathematical object with restriction maps that define how to glue τover local neighborhoods $U \subseteq K$, discussed in section **??**, such that the A maps are consistent over continuous regions of K. While Acan usually construct graphical elements solely with the data in τ, some visualizations, such as line, may also need some finite number n of derivatives, which is captured by the jet bundle $\mathcal{J}^n$ [28, 29] with $\mathcal{J}^0(E) = E$. In this work, we at most need $\mathcal{J}^2(E)$ which is the value at τ and its first and second derivatives; therefore the artist takes as input the jet bundle $E' = \mathcal{J}^2(E)$.

Specifically, $A$is the equivariant map from E' to a specific graphic $\rho \in \Gamma(H)$

$$\begin{tikzcd}
E' \arrow[r, "\nu"] \arrow[dr, "\pi"'] & V \arrow[d, "\pi"] & \xi^*V \arrow[l, "\xi^*"'] \arrow[r, "Q"] \arrow[d, "\xi^*\pi"'] & H \arrow[dl, "\pi"] \\
& K & S \arrow[l, "\xi"] &
\end{tikzcd} \tag{22}$$

where the input can be point wise $\tau(k) \mid k \in K$. The encoders $\nu : E' \to V$ convert the data components to visual components(1.2.2). The continuity map $\xi : S \to K$ then pulls back the visual bundle V over S(1.3.2). Then the assembly function $Q : \xi^*V \to H$ composites the fiber components of ξ^*V into a graphic in H(1.3.3). This functional decomposition of the visualization artist facilitates building reusable components at each stage of the transformation because the equivariance constraints are defined on ν, Q, and ξ.

1.3.1 Visual Fiber Bundle V

We introduce a visual bundle V to store the visual representations the artist needs to assemble into a graphic. The visual bundle (V, K, π, P) has section $\mu : V \to K$ that resolves to a visual variable in the fiber P. The visual bundle V is the latent space of possible parameters of a visualization type, such as a scatter or line plot. We define P in terms of the parameters of a visualization libraries compositing functions; for example table 1 is a sample of the fiber space for Matplotlib [30].

$\boldsymbol{\nu_i}$	$\boldsymbol{\mu_i}$	$\boldsymbol{codomain(\nu_i) \subset P_i}$
position	x, y, z, theta, r	$\mathbb{R}$
size	linewidth, markersize	$\mathbb{R}^+$
shape	markerstyle	$\{f_0, \ldots, f_n\}$
color	color, facecolor, markerfacecolor, edgecolor	$\mathbb{R}^4$
texture	hatch	$\mathbb{N}^{10}$
	linestyle	$(\mathbb{R}, \mathbb{R}^{+^{n, n\%2=0}})$

Table 1: Some possible components of the fiber P for a visualization function implemented in Matplotlib

A section μ is a tuple of visual values that specifies the visual characteristics of a part of the graphic. For example, given a fiber of $\{xpos, ypos, color\}$ one possible section could be $\{.5, .5, (255, 20, 147)\}$. The $codomain(\nu_i)$ determines the monoid actions on P_i. These fiber components are implicit in the library, by making them explicit as components of the fiber we can build consistent definitions and expectations of how these parameters behave.

1.3.2 Visual Encoders ν

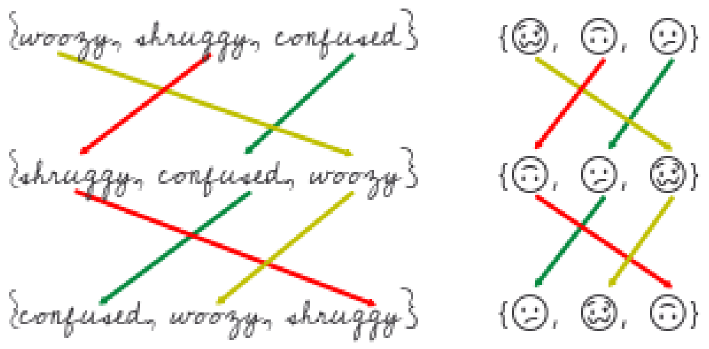
{woozy, shruggy, confused}
{shruggy, confused, woozy}
{confused, woozy, shruggy}

Figure 8: In this artist, ν maps the strings to the emojis. This ν is equivariant because the monoid actions (which are represented by the colored arrows) are the same on both the τ input and μ output sets.

As introduced in section **??**, there are many ways to visually represent data components. We define the visual transformers ν

$$\{\nu_0, \dots, \nu_n\} : \{\tau_0, \dots, \tau_n\} \mapsto \{\mu_0, \dots, \mu_n\} \tag{23}$$

as the set of equivariant maps $\nu_i : \tau_i \mapsto \mu_i$. Given M_i is the monoid action on E_i and that there is a monoid $M_i{}'$ on V_i, then there is a monoid homomorphism from $\varphi : M_i \to M_i{}'$ that ν must preserve. As mentioned in section 1.1.2, we choose monoid actions as the basis for equivariance because they define the structure on the fiber components.

A validly constructed ν is one where the diagram of the monoid transform m commutes such that

$$\begin{tikzcd} E_i \arrow[r, "\nu_i"] \arrow[d, "m_r"'] & V_i \arrow[d, "m_v"] \\ E_i \arrow[r, "\nu_i"] & V_i \end{tikzcd} \tag{24}$$

In general, the data fiber F_i cannot be assumed to be of the same type as the visual fiber P_i and the actions of M on F_i cannot be assumed to be the same as the actions of M' on P; therefore an equivariant ν_i must satisfy the constraint

$$\nu_i(m_r(E_i)) = \varphi(m_r)(\nu_i(E_i)) \tag{25}$$

such that φ maps a monoid action on data to a monoid action on visual elements. However, we can construct a monoid action of M on P_i that is compatible with a monoid action of M on F_i. We can compose the monoid actions on the visual fiber $M' \times P_i \to P_i$ with the homomorphism φ that takes M to M'. This allows us to define a monoid action on P of M

that is $(m, v) \to \varphi(m) \bullet v$. Therefore, without a loss of generality, we can assume that an action of M acts on F_i and on P_i compatibly such that φ is the identity function.

On example of an equivariant ν is illustrated in figure 8, which is a mapping from **Strings** to symbols. The data is an example of a Steven's nominal measurement set, which is defined as having on it permutation group actions

$$\text{if } r_1 \neq r_2 \text{ then } \nu(r_1) \neq \nu(r_2) \tag{26}$$

such that shuffling the words must have an equivalent shuffle of the symbols they are mapped to. This is illustrated in the identical actions, represented by the colored arrows, on the words and emojis. To preserve ordinal and partial order monoid actions, ν must be a monotonic function such that given $r_1, r_2 \in E_i$,

$$\text{if } r_1 \leq r_2 \text{ then } \nu(r_1) \leq \nu(r_2) \tag{27}$$

the visual encodings must also have some sort of ordering. For interval scale data, ν is equivariant under translation monoid actions if

$$\nu(x + c) = \nu(x) + c \tag{28}$$

while for ratio data, there must be equivalent scaling[14]

$$\nu(xc) = \nu(x) * c \tag{29}$$

We therefore can test if a ν is equivariant by testing the actions under which is must commute. For example, we define a transform $\nu_i(x) = .5$ on interval data. This means it must commute under translation, for example $t(x) = x + 2$. Testing this constraint

$$\nu(t(r + 2)) \stackrel{?}{=} \nu(r) + 2 \tag{30}$$

$$.5 \neq .5 + 2 \tag{31}$$

we find that the ν defined here does not commute and is therefore invalid. The constraints on ν can be embedded into our artist such that the ν functions can test for equivariance and also provide guidance on constructing new ν functions.

1.3.3 Graphic Assembler Q

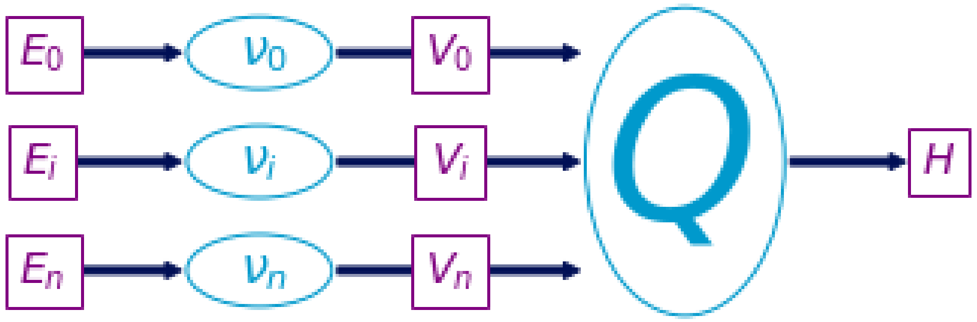

Figure 9: ν_i functions convert data τ_i to visual characteristics μ_i, then Q assembles μ_i into a graphic ρ such that there is a map ξ preserving the continuity of the data. ρ applied to a region of connected components S_j generates a part of a graphic, for example the point graphical mark.

As shown in figure 9, the assembly function Q combines the fiber F_i wise ν transforms into a graphic in H. Together, ν and Q are a map-reduce operation: map the data into their visual encodings, reduce the encodings into a graphic. As with ν the constraint on Q is that for every monoid action on the input μ there is corresponding monoid action on the output ρ.

While ρ generates the entire graphic, we will restrict the discussion of Q to generation of sections of a glyph. We formally describe a glyph as Q applied to the regions k that map back to a set of path connected components $J \subset K$ as input:

$$J = \{j \in K \text{ s. t. } \exists \gamma \text{ s.t. } \gamma(0) = k \text{ and } \gamma(1) = j\} \tag{32}$$

where the path[31] γ from k to j is a continuous function from the interval [0,1]. We define the glyph as the graphic generated by $Q(S_j)$

$$\begin{tikzcd} H \arrow[r, shift left] & S_j \arrow[l, shift left, "\rho(S_j)"] \arrow[r, shift left, "\xi(s)"] & J_k \arrow[l, shift left, "\xi^{-1}(J)"] \end{tikzcd} \tag{33}$$

such that for every glyph there is at least one corresponding region on K. This is in keeping with the definition of glyph as any differentiable element put forth by Ziemkiewicz and Kosara[32]. The primitive point, line, and area marks[33, 34] are specially cased glyphs.

It is on sections of these glyphs that we define the equivariant map as $Q : \mu \mapsto \rho$ and an action on the subset of graphics $Q(\Gamma(V)) \in \Gamma(H)$ that Q can generate. We then define the constraint on Qsuch that if Q is applied to μ, μ' that generate the same ρ then the output of both sections acted on by the same monoid m must be the same. While it may seem intuitive that visualizations that generate the same glyph should consistently generate the same glyph given the same input, we formalize this constraint such that it can be specified as part of the implementation of Q.

Lets call the visual representations of the components $\Gamma(V) = X$ and the graphic $Q(\Gamma(V)) = Y$. If for elements of the monoid $m \in M$ and for all $\mu, \mu' \in X$, we define the monoid action on X so that it is by definition equivariant

$$Q(\mu) = Q(\mu') \implies Q(m \circ \mu) = Q(m \circ \mu') \tag{34}$$

then a monoid action on Y can be defined as $m \circ \rho = \rho'$. The transformed graphic ρ' is equivariant to a transform on the visual bundle $\rho' = Q(m \circ \mu)$ on a section that $\mu \in Q^{-1}(\rho)$ that must be part of generating ρ.

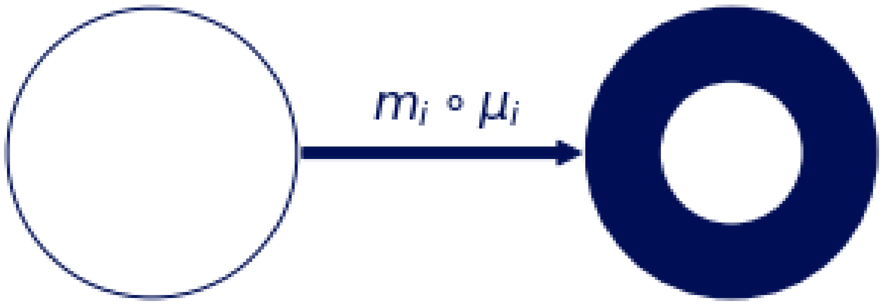

Figure 10: These two glyphs are generated by the same Q function. The monoid action m_i on edge thickness μ_i of the first glyph yields the thicker edge $\mu_i{}'$ in the second glyph.

The glyph in figure 10 has the following characteristics P specified by ($xpos$, $ypos$, $color$, $thickness$) such that one section is $\mu = (0, 0, 0, 1)$ and $Q(\mu) = \rho$ generates a piece of the thin hollow circle. The equivariance constraint on Q is that the action $m = (e, e, e, x + 2)$, where e is identity, translates μ to $\mu' = (e, e, e, 3)$. The corresponding action on ρ causes $Q(\mu')$ to be the thicker circle in figure 10.

1.3.4 Assembly Q

In this section we formulate the minimal Q that will generate distinguishable graphical marks: non-overlapping scatter points, a non-infinitely thin line, and an image.

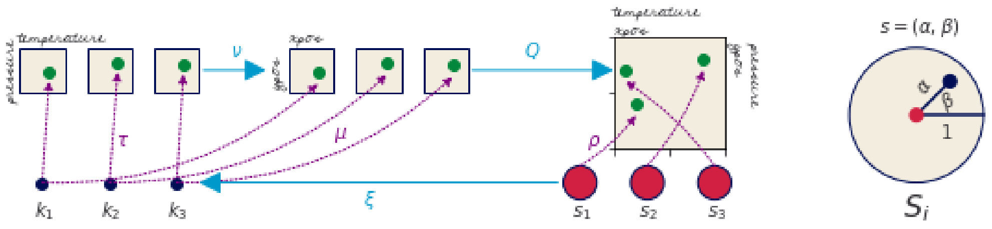

Figure 11: The data is discrete points (temperature, time). Via ν these are converted to (xpos, ypos) and pulled over discrete S. These values are then used to parameterize ρ which returns a color based on the parameters (xpos,ypos) and position α, β on S_k that ρ is evaluated on.

The scatter plot in figure 11 can be defined as $Q(xpos, ypos)(\alpha, \beta)$ where color $\rho_{RGB} = (0, 0, 0)$ is defined as part of Q and $s = (\alpha, \beta)$ defines the region on S. The position of this swatch of color can be computed relative to the location on the disc S_k as shown in figure 11:

$$x = size * \alpha \cos(\beta) + xpos \tag{35}$$
$$y = size * \alpha \sin(\beta) + ypos \tag{36}$$

such that $\rho(s) = (x, y, 0, 0, 0)$ colors the point (x,y) black. Here $size$ can either be defined inside Q or it could also be a parameter in V that is passed along with (xpos, vpos). As seen in figure 11, a scatter has a direct mapping from a region on S_k to its corresponding k.

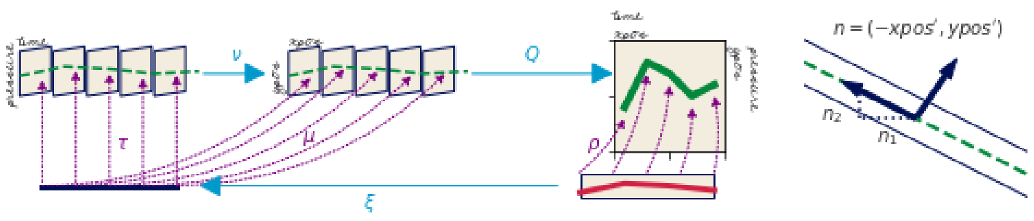

Figure 12: The line fiber $(time, temp)$ is thickened with the derivative $(time', temperature'$ because that information will be necessary to figure out the tangent to the point to draw a line. This is because the line needs to be pushed perpendicular to the tangent of (xpos, ypos). The data is converted to visual characteristics (xpos, ypos). The α coordinates on S specifies the position of the line, the β coordinate specifies thickness.

In contrast to the scatter, the line plot $Q(xpos, \hat{n_1}, ypos, \hat{n_2})(\alpha, \beta)$ shown in fig 12 has a ξ function that is not only parameterized on k but also on the α distance along k and corresponding region in S.The line also exemplifies the need for the jet since the line needs to know the tangent of the data to draw an envelope above and below each (xpos,ypos) such that the line appears to have a thickness. The magnitude of the slope is

$$|n| = \sqrt{{n_1}^2 + {n_2}^2} \tag{37}$$

such that the normal is

$$\hat{n_1} = \frac{n_1}{|n|},\ \hat{n_2} = \frac{n_2}{|n|} \tag{38}$$

which yields components of ρ

$$x = xpos(\xi(\alpha)) + width * \beta \hat{n_1}(\xi(\alpha)) \tag{39}$$
$$y = ypos(\xi(\alpha)) + width * \beta \hat{n_2}(\xi(\alpha)) \tag{40}$$

where (x,y) look up the position $\xi(\alpha)$ on the data. At that point, we also look up the the derivatives $\hat{n_1}, \hat{n_2}$ which are then multiplied by a *width* parameter to specify the thickness. As with the *size* parameter in scatter, *width* can be defined in Q or as a component of V.

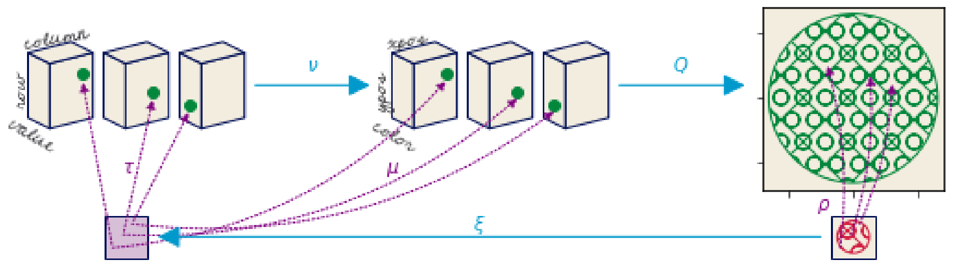

Figure 13: The only visual parameter an image requires is color since ξ encodes the mapping between position in data and position in graphic.

The image $Q(xpos, ypos, color)$ in figure 13 is a direct lookup into $\xi : S \to K$. Since K is 2D continuous space, the indexing variables (α, β) define the distance along the space. This is then used by ξ to map into K to lookup the color values

$$R = R(\xi(\alpha, \beta)) \tag{41}$$
$$G = G(\xi(\alpha, \beta)) \tag{42}$$
$$B = B(\xi(\alpha, \beta)) \tag{43}$$

that the data values have been mapped into. In the case of an image, the indexing mapper ξ may do some translating to a convention expected by Q, for example reorientng the array such that the first row in the data is at the bottom of the graphic.

1.3.5 Assembly factory $\hat{Q}$

The graphic base space S is not accessible in many architectures, including Matplotlib; instead we can construct a factory function $\hat{Q}$ over K that can build a Q. As shown in eq 22, Q is a bundle map $Q : \xi^*V \to H$ where ξ^*V and H are both bundles over S.

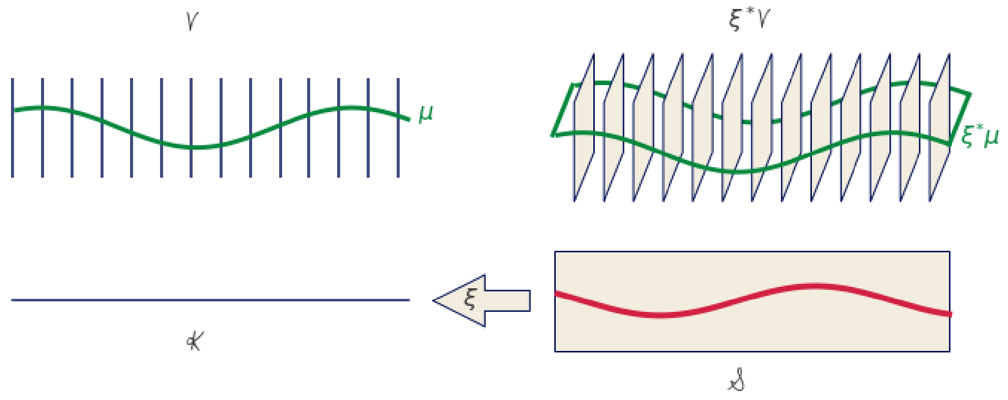

Figure 14: The pullback of the visual bundle ξ^*V is the replication of a μ over all points s that map back to a single k. Because the μ is the same, we can constract a $\hat{Q}$ on μ over k that will fabricate the Q for the equivalent region of s associated to that k

The preimage of the continuity map $\xi^{-1}(k) \subset S$ is such that many graphic continuity points $s \in S_K$ go to one data continuity point k; therefore, by definition the pull back of μ

$$\xi^*V\mid_{\xi^{-1}(k)} = \xi^{-1}(k) \times P \tag{44}$$

copies the visual fiber P over the the points s in graphic space S that correspond to one k in data space KṪhis set of points s are the preimage $\xi^{-1}(k)$ of k.

This copying is illustrated in figure 14, where the 1D fiber $P \hookrightarrow V$ over K is copied repeatedly to become the 2D fiber $P^*\mu \hookrightarrow \xi^*V$ with identical components over S. Given the section $\xi^*\mu$ pulled back from μ and the point $s \in \xi^{-1}(k)$, there is a direct map from μon a point k, there is a direct map from the visual section over data base space $(k, \mu(k)) \mapsto (s, \xi^*\mu(s))$ to the visual section $\xi^*\mu$ over graphic base space. This map means that the pulled back section $\xi^*\mu(s) = \xi^*(\mu(k))$ is the section μ copied over all s. This means that $\xi^*\mu$ is identical for all s where $\xi(s) = k$, which is illustrated in figure 14 as each dot on $P$is equivalent to the line intersection $P^*\mu$.

Given the equivalence between μ and $\xi^*\mu$ defined above, the reliance on S can be factored out. When Q maps visual sections into graphics $Q : \Gamma(\xi^*V) \to \Gamma(H)$, if we restrict Q input to the pulled back visual section $\xi^*\mu$ then

$$\rho(s) := Q(\xi^*\mu)(s) \tag{45}$$

the graphic section ρ evaluated on a visual region s is defined as the assembly function Q with input pulled back visual section $\xi^*\mu$ also evaluated on s. Since the pulled back visual section $\xi^*\mu$ is the visual section μ copied over every graphic region $s \in \xi^{-1}(k)$, we can define a Q factory function

$$\hat{Q}(\mu(k))(s) := Q((\xi^*\mu)(s)) \tag{46}$$

where the assembly function $\hat{Q}$ that takes as input the visual section on data μ is defined to be the assembly function Q that takes as input the copied section $\xi^*\mu$ such that both functions are evaluated over the same location $\xi^{-1}(k) = s$ in the base space S.

Factoring out s from equation 46 yields $\hat{Q}(\mu(k)) = Q(\xi^*\mu)$ where Q is no longer bound to input but $\hat{Q}$ is still defined in terms of K. In fact, $\hat{Q}$ is a map from visual space to graphic space $\hat{Q} : \Gamma(V) \to \Gamma(H)$ locally over k such that it can be evaluated on a single visual record $\hat{Q} : \Gamma(V_k) \to \Gamma(H \mid_{\xi^{-1}(k)})$. This allows us to construct a $\hat{Q}$ that only depends on K, such that for each $\mu(k)$ there is part of $\rho \mid_{\xi^{-1}(k)}$. The construction of $\hat{Q}$ allows us to retain the functional map reduce benefits of Q without having to majorly restructure the existing pipeline for libraries that delgate the construction of ρ to a back end such as Matplotlib.

1.3.6 Sheafs

The restriction maps of a sheaf describe how local τcan be glued into larger sections [35, 36]. As part of the definition of local triviality, there is an open neighborhood $U \subset K$ for every $k \in K$. We can define the inclusion map $\iota : U \to K$ which pulls Eover U

$$\begin{tikzcd} \iota^* E \arrow[r, hook, "\iota^*"] \arrow[d, "\pi"'] & E \arrow[d, "\pi"'] \\ U \arrow[r, hook, "\iota"] \arrow[u, bend right, "\iota^*\tau"'] & K \arrow[u, bend right, "\tau"'] \end{tikzcd} \tag{47}$$

such that the pulled back $\iota^*\tau$ only contains records over $U \subset K$. By gluing $\iota^*\tau$ together, the sheaf is putting a continuous structure on local sections which allows for defining a section over a subset in K. That section over subset K maps to the graphic generated by A for visualizations such as sliding windows[37, 38] streaming data, or navigation techniques such as pan and zoom[39].

1.3.7 Composition of Artists: $+$

To build graphics that are the composites of multiple artists, we define a simple addition operator that is the disjoint union of fiber bundles E. For example, a scatter plot E_1 and a line plot E_2 have different Kthat are mapped to separate S. To fully display both graphics, the composite graphic $A_1 + A_2$ needs to include all records on both K_1 and K_2, which are the sections on the disjoint union $K_1 \sqcup K_2$. This in turn yields disjoint graphics $S_1 \sqcup S_2$ rendered to the same image. Constraints can be placed on the disjoint union such as that the fiber components need to have the same νposition encodings or that the position μneed to be in a specified range. There is a second type of composition where E_1 and E_2 share a base space $K_2 \hookrightarrow K_1$ such that the the artists can be considered to be acting on different components of the same section. This type of composition is important for creating visualizations where elements need to update together in a consistent way, such as multiple views [40, 41] and brush-linked views[42, 43].

1.3.8 Equivalance class of artists A'

It is impractical to implement an artist for every single graphic; instead we implement an approximation of an the equivalence class of artists $\{A \in A' : A_1 \equiv A_2\}$. Roughly, equivalent artists have the same fiber bundle V and same assembly function Q but act on different sections μ, but we will formalize the definition of the equivalance class in future work. As a first pass for implementation purposes, we identify a minimal P associated with each A' that defines what visual characteristics of the graphic must originate in the data such that the graphic is identifiable as a given chart type.

For example, a scatter plot of red circles is the output of one artist, a scatter plot of green squares the output of another. These two artists are equivalent since their only difference is in the literal visual encodings (color, shape). Shape and color could also be defined in Q but the position must come from the fiber $P = (xpos, ypos)$ since fundementally a scatter plot is the plotting of one position against another[44]. We also use this criteria to identify derivative types, for example the bubble chart[45] is a type of scatter where by definition the glyph size is mapped from the data. The criteria for equivalence class membership serves as the basis for evaluating invariance[**kindlmann2014algebraic**].