1 Topological Artist Model

can change this all to 3rd person passive later
In this section we introduce a mathematical description of the visualization pipeline where artist $\mathscr{A}$ functions transform data space $\mathscr{E}$ to an intermediate representation in a prerendered graphic space $\mathscr{H}$.

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

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

We build on Butler's proposal of using fiber bundles as a common data representation format for visualization data[4, 5] because fiber bundles are mathematical structures that are flexible enough express all the types of data described in section **??**.

We model data as the fiber bundle (E, B, π, F), where E, F, and K are topological spaces that encode

- the properties of the variables in the fiber F (1.1.1)
- the continuity of the records in the base space K (1.1.3)
- collections of records τ (1.1.4).

and E is the total space of data that F lives in. The bundle is the projection map π

$$F \hookrightarrow E \xrightarrow{\pi} K \tag{2}$$

that binds the variable space F to the base space K. The fiber bundles mentioned in this work are assumed to be trivial[12, 19], unless otherwise specified, because the trivial bundle is $E = K \times F$ such that extra structure in the total space E falls out and discussion can be focused on the fiber and base space.

1.1.1 Variables: Fiber Space F

yes, I know i may need to change this U to like $\mathcal{U}$, spivak uses U The fiber is a topological space that is the set of possible values of the data; the values themselves can be any dimension and type and have any continuity. We use Spivak's description of simplicial database schemas [20] as the basis of our fiber space because he binds the components of the fiber to variable names and types. Spivak constructs a set U that is the disjoint union of all possible objects of types $\{T_0, \ldots, T_n\} \in DT$ where DT are the data types of the variables in the dataset. He then defines the single variable set U_σ

$$\begin{array}{ccc} U_\sigma & \longrightarrow & U \\ {\scriptstyle \pi_\sigma}\downarrow & & \downarrow{\scriptstyle \pi} \\ C & \xrightarrow[\sigma]{} & \mathbf{DT} \end{array} \tag{3}$$

which is U restricted to objects of type T bound to variable name c. Given U_σ, the fiber for a one variable dataset is

$$F = U_{\sigma(c)} = U_T \tag{4}$$

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 U_σ applied to all the columns:

$$F = U_{T_1} \times \ldots UT_i \ldots \times U_{T_n} \tag{5}$$

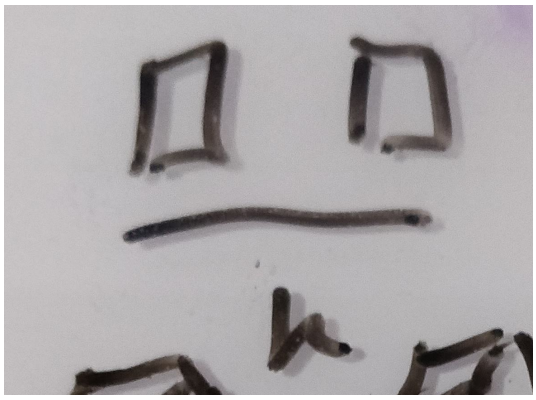
which is equivalent to

$$F = F_0 \times \ldots \times F_i \times \ldots \times F_n \tag{6}$$

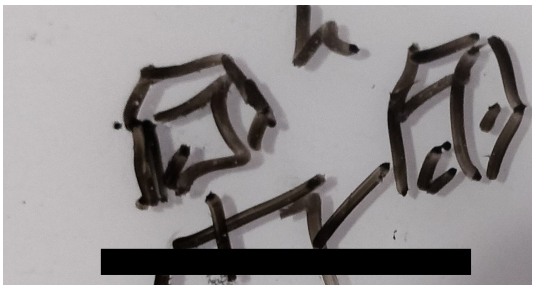
which allows us to decouple F into its component fibers F_i. For every variable

$$V_i = (c_i, T_i, U_{T_i}) \tag{7}$$

we keep track of its name c, type T, and values U_T.



(a)



(b)

Figure 1: These two datasets have the same base space K, but figure 1a has fiber $F = \mathbb{R}\times\mathbb{R}^+$ which is (time, temperature) while figure 1b has fiber $\mathbb{R}^+ \times \mathbb{R}^2$ which is (time, wind=(speed, direction))

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

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

where $V_0 = (time, \texttt{datetime}, \mathbb{R}^+)$ and $V_1 = (temperature, \texttt{float}, \mathbb{R})$. 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 $V_1 = (wind, \texttt{wind}, \mathbb{R}^2)$

1.1.2 Measurement Scales: Monoid Actions

After specifying the variables V in the dataset, we next describe ways in which we can transform the variables by identifying the monoid actions M_i on the F_i. We use monoids as the abstraction because they encode composibilty, which maps well to the data transformation process in a software library [25].

A monoid [13] M_i is a set with an associative binary operator $* : M_i \times M_i \to M_i$. A monoid has an identity element $e \in M_i$ such that $e * a = a * e = a$ for all $a \in M_i$. A left monoid action [1, 18] of M_i is a set F_i with an action $\bullet : M \times F_i \to F_i$ with the properties:

$$\begin{aligned} \textbf{associativity} \text{ for all } f, g \in M_i \text{ and } x \in F_i,\ & f \bullet (g \bullet x) = (f * g) \bullet x \\ \textbf{identity} \text{ for all } x \in F_i, e \in M_i,\ & e \bullet x = x \end{aligned}$$

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 the monoid M_i on each $F_i \in F$. We expand our definition of variables

$$V_i = (c_i,\, T_i,\, F_i\, M_i) \tag{11}$$

to include the monoid actions that act on F_i.

Steven's described the measurement scales[11, 22] 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 [24]. For example, given $V_0 = (\textit{temperature}, \texttt{float}, \mathbb{R})$ which is interval data:

- monoid operator addition $* = +$
- monoid operations: $f : x \mapsto x + c_1$, $g : x \mapsto x + c_2$
- monoid action operator composition $\bullet = \circ$

then the translation monoid actions on temperature satisfy the condition

$$\begin{array}{ccc} \mathbb{R} & & \\ {\scriptstyle x+c_1}\downarrow & \searrow^{(x+c_1)\circ(x+c_2)} & \\ \mathbb{R} & \xrightarrow[x+c_2]{} & \mathbb{R} \end{array} \tag{12}$$

where c_1, c_2 are intervals $x_i - x_j$.

1.1.3 Continuity: Base Space K

The advantage of fiber bundles is they provide a way to encode the continuity in a dataset as the basespace K without making assumptions as to what that continuity is. In turn this representation of continuity can then be used to keep track of how the data fits together, for example if a visualization of a very large dataset calls for parallelization.

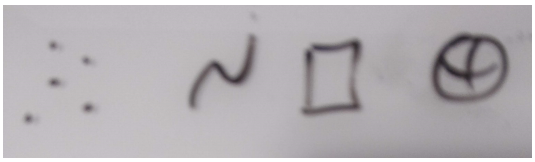


Figure 2: The topological base space K encodes the connectivity of the data space, for example if the data is independent points or a map or on a sphere

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 be continuous or discrete.

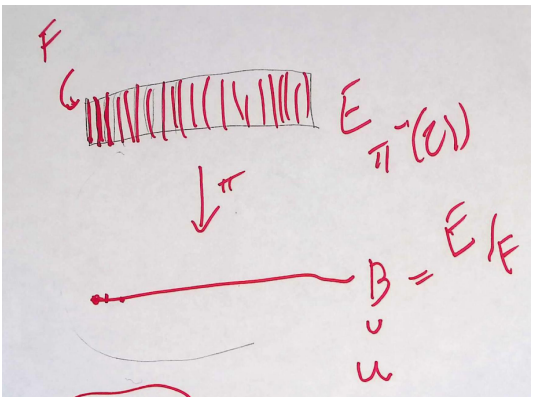


Figure 3: The base space E is divided into fiber segments F. The base space K acts as an index into the records in the fibers. this figure might be good all the way up top to lay out the components of fb

Formally K is the quotient space [16] of E, meaning it is the finest space[2] such that every $k \in K$ has a corresponding fiber F_k in E[16]. In figure 3, E is a rectangle divided by vertical fibers F, so the minimal K for which there is always a mapping $\pi : F \to K$ is the line.

As with fibers and monoids, we can decompose the total space into components $\pi : E_i \to K$ where

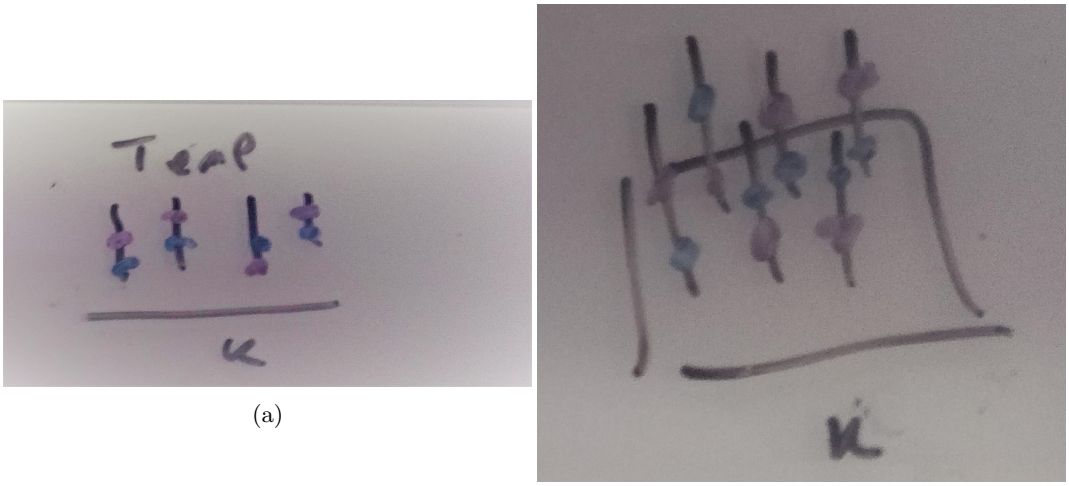
$$\pi : E_1 \oplus \ldots \oplus E_i \oplus \ldots \oplus E_n \to K \tag{13}$$

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.

The datasets in figure 4 have the same fiber of (temperature, time) which is $F = \mathbb{R} \times \mathbb{R}^+$. 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.

1.1.4 Data: Sections τ

While the fiber and base space describe the general structure of all data that lives in the fiber bundle, the sections $\tau : K \to E$ define the datasets that live in the fiber. We generalize



(a)

(b)

Figure 4: These two datasets have the same (time, temperature) fiber. In figure **??** the total space E is discrete over points $k \in K$, meaning the records in the fiber are also discrete. In figure **??** E lies over the continuous interval K, meaning the records in the fiber are sampled from a continous space. revamp figure: F=Plane, k1 = dots, k2=line

Spivak's section as table τ[20] to any sort of structured dataset such that

$$\begin{array}{ccc} F & \hookrightarrow & E \\ & & \pi \downarrow \uparrow \tau \\ & & K \end{array} \tag{14}$$

such that there is $\pi(\tau(k)) = k$ map back to K. There can be many sections τ, and the space of global sections is $\Gamma(E)$. For a trivial fiber bundle

$$\tau(k) = (k, (g_{F_0}(k), \ldots, g_{F_n}(k)) \tag{15}$$

where $g : K \to F$ is the index function into the fiber. Because we can decompose the bundle and the fiber, we can formulate τ as

$$\tau = (\tau_0, \ldots, \tau_i, \ldots, \tau_n) \tag{16}$$

where each section τ_i is a variable or set of variables.

In the example in figure 5, the fiber is $\mathbb{R}^+ \times \mathbb{R}$ as described in figure 1 and the base space is the interval K. The section τ_1 resolves to a series of monotonically increasing in time records of temperature. Section τ_2 returns a different timeseries of temperature values. Both sections are included in the global set of sections $\tau_1, \tau_2 \in \Gamma(E)$.

1.1.5 Sheaf and Stalk

Often a graphic may need to be updated with live data or support zooming in on a segement of the dataset; to support working with a subset of data, we can use the sheaf $\mathcal{O}(E)$. All fiber

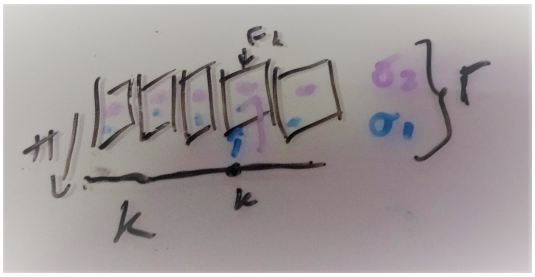


Figure 5: Fiber (time, temperature) with an interval K basespace. The sections τ_1 and τ_2 are constrained such that the time variable must be monotonic, which means each section is a timeseries of temperature values. They are included in the global set of sections $\tau_1, \tau_2 \in \Gamma(E)$

bundles are locally trivial, which means that E restricted over a small enough neighborhood $U \subset K$ is a locally trivial bundle over U[12]. The sheaf $O(E)$ is the localized section of fibers $\iota^*\tau : U \to \iota^*E$

$$\begin{array}{ccc} \iota^*E & \xleftarrow{\iota^*} & E \\ {\scriptstyle\pi}\downarrow\uparrow{\scriptstyle\iota^*\tau} & & {\scriptstyle\pi}\downarrow\uparrow{\scriptstyle\tau} \\ U & \xleftarrow{\iota} & K \end{array} \tag{17}$$

pulled back over the neighborhood U via the inclusion map $\iota : U \to K$. The localized section is the germ $\xi^*\tau$. The neighborhood of points k_i surrounding the point k the sheaf lies over is the stalk $\mathcal{F}_b$ [19, 21]. While E is only the records in a fiber F_k over a specific k, because the stalk includes the neighborhood U it also includes nearby records. While this can be useful for visual transforms, often the extra needed information can be found in the jet bundle $\mathcal{J}$ [10, 15]. For example, line thickness requires the derivative of the given position to be rendered, which can be found in $E' = E + \mathcal{J}(E)$

1.2 Graphic: H

We can separate the structure of the graphic from the properties of the output format by modeling the space of graphics as a fiber bundle (H, S, π, D). As with data, the fiber bundle is for a class of graphics with shared base space S (1.2.1) and fiber D (1.2.2) and the sections ρ (1.2.3) encode the fully specified graphics.

1.2.1 Abstract Display D

The fiber D is specified in terms of the target display space, for example a 2D screen or 3D printer. In this work, we assume a 2D opaque image $D = \mathbb{R}^5$ with elements in D of the form

$$d = (x, y, r, g, b) \tag{18}$$

such that a rendered graphic only consists of 2D position and color. To support overplotting and transparency, the fiber could be $D = R^7$ such that $d = (x, y, z, r, g, b, a)$. The location coordinates x and y are defined in terms of the display, while the (r, g, b) values are filled in via a lookup on S.

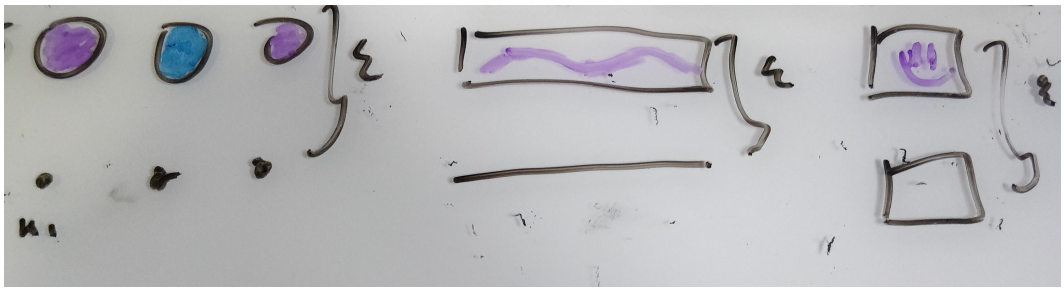


Figure 6: The scatter and line graphic base spaces S are larger than K so that they can encode physical aspects of the glyph, such as shape (a circle) or thickness. The heatmap base S is the same as K because a heatmap is directly mapping the values in the data to the graphic. add α, β coordinates

1.2.2 Continuity of the Graphic S

An assumption of graphical representations is that they match the continuity of the data[8, 23], but the underlying topology S of a graphic may need more dimensions than the data topology K so that the glyph can be defined in D. Therefore we define the base space mapping from graphic S to data K

$$\begin{array}{ccc} E & & H \\ {\scriptstyle\pi}\downarrow & & {\scriptstyle\pi}\downarrow \\ K & \xleftarrow{\xi} & S \end{array} \tag{19}$$

as the deformation retraction [17] $\xi : S \to K$ that goes from a region $s \in S_k$ to its associated point k, such that when $\xi(s) = k$, $\xi^*\tau(s) = \tau(k)$. While dimensions can be added to S, it retains the same continuity as K.

In figure 6 each disk S_k indexes how elements in D are held together to generate a single glyph that is the visual representation of a single point in F_k. For the line, the region β over a point α_i specifies the thickness of the line in D for the corresponding section on the interval K. The heatmap has the same continuity in data space and graphic space such that no extra dimensions are needed.

1.2.3 Renderable Glyph ρ

A section $\rho : S \to H$ defines a piece of the graphical representation of the data. The physical display coordinates in D are not equivalent to the mathematical coordinates in S such that many points $s \in S$ might map to a single $d \in D$ and many d might go into a visible section of a displayed graphic. Therefore, we define the bounding box $b = [y_{top}, y_{bottom}, x_{right}, x_{left}]$ in the display space D that is a piece of the rendered graphic.

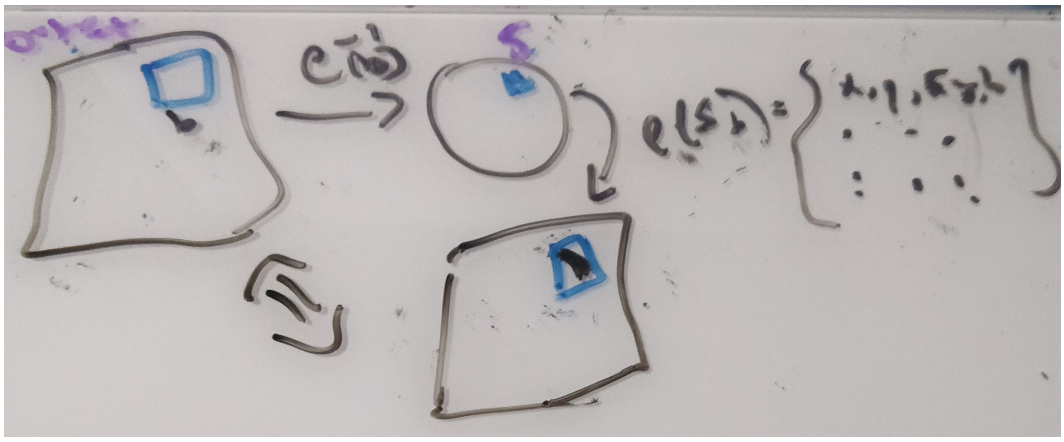


Figure 7: To render a graphic, a region b is selected in the display space. This region is in coordinates D so the inverse mapping $\rho^{-1}(b)$ maps to an equivalent region $S_b \subset S$. We then take the sections $\rho(S_b)$ which are the elements $\{d_0, \ldots, d_n\}$ that make the piece of the graphic in b.

We inverse map this bounding box $S_b = \rho^{-1}(b)$ into a region $S_b \subset S$

$$R_b = \iint_{S_b} \rho_r(s) ds^2 \tag{20}$$

$$G_b = \iint_{S_b} \rho_g(s) ds^2 \tag{21}$$

$$B_b = \iint_{S_b} \rho_b(s) ds^2 \tag{22}$$

to then compute the remaining (R, G, B) components of the bounded display region D_b.

As shown in figure 7, the output space queries into the graphic bundle to render the image. We select a region b in the output space, inverse map the region into $S_b \subset S$, then take the section ρ over the region S_b. The section yields the set of elements $\{d_0, \ldots, d_n\}$ that specify the (r, g, b) values corresponding to the coordinates in b.

1.3 Artist

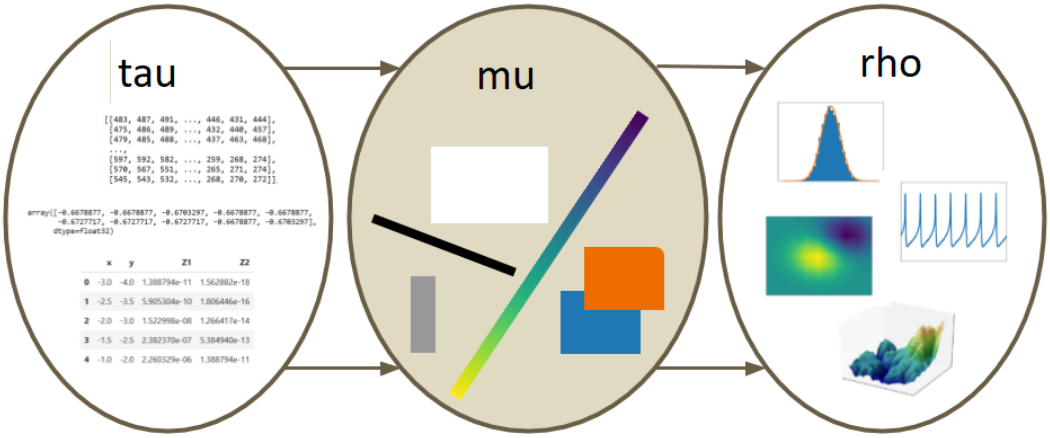


Figure 8: The artist A converts an section of data τ into a section of visual equivalents μ that are then composed into a piece of a glyph defined by the graphic section ρ might need to label this as E', V, H

The artist is the function that converts data into graphics; its name is taken from the analogous part of the existing Matplotlib architecture that builds visual elements to pass off to the renderer. The artist is a mapping from E padded with data from $\mathcal{J}(E)$ to a graphic that is a section ρ in $\Gamma(H)$

$$\begin{array}{ccccccc} E' & \xrightarrow{\nu} & V & \xleftarrow{\xi^*} & \xi^*V & \xrightarrow{Q} & \Gamma(H) \\ & \searrow^{\pi} & \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \xi^*\pi} & \swarrow_{\pi} & \\ & & K & \xleftarrow{\xi} & S & & \end{array} \tag{23}$$

with an intermediate fiber bundle V to hold visual representations and stages

1. ξ binding the continuity in the graphic to the continuity in the data (1.2.2)
2. ν conversion of data into visual characteristics (1.3.2)
3. Q assembly of visual variables into a glyph (1.3.3)

of the visual transformation illustrated in figure 8. The functions ξ, ν and Q are constructed such that the input can be a single section on K or S, which allows the artist to be trivially parallelized or formulated as a delayed computer graph or otherwise not need to immediately work on all the data.

1.3.1 Visual Fiber Bundle V

The visual fiber bundle (V, K, π, P) has section $\mu : V \to K$ that resolves to a visual variable [3, 14] in fiber P. The fiber space P is defined in terms of the parameters of the visualization specification and the values those parameters resolve to - for example APT[**mackinlayAutomatingDesignGraphical1986**] or Vega[**satyanarayanDeclarativeInteractionDesign2014**

$\boldsymbol{\nu_i}$	$\boldsymbol{\mu_i}$	$\boldsymbol{codomain(\nu_i)}$
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	$\{f_0, \ldots, f_n\} \times (\mathbb{R}, \mathbb{R}^{+^{n, n\%2=0}})$

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

Table **??** is a sample of the fiber space for Matplotlib [9]. A section μ of V 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 μ_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 Channels

```
[2]: nu = {'confused':'😕', 'woozy':'😵', 'shruggy':'🙃'}

[3]: nu.keys()

[3]: dict_keys(['confused', 'woozy', 'shruggy'])

[4]: nu.values()

[4]: dict_values(['😕', '😵', '🙃'])

14]: values

14]: ['woozy', 'shruggy', 'confused']

15]: [nu[v] for v in values]

15]: ['😵', '🙃', '😕']
```

Figure 9: An equivariant ν preserves monoid actions. Here ν maps the words to the emojis and is constructed such that a permutation of the words is reflected in a permutation of the symbols. will switch to a figure table thing but this is the idea

As introduced in section **??**, there are many ways to encode data visually. We define the visual transformers ν as the set of conversion functions

$$\{\nu_0, \ldots, \nu_n\} : \{\tau_0, \ldots, \tau_n\} \mapsto \{\mu_0, \ldots, \mu_n\} \tag{24}$$

such that $\nu_i : \tau_i \mapsto \mu_i$ is an equivariant map such that there is a monoid homomorphism from F_i to V_i. A validly constructed ν is one where the diagram

$$\begin{array}{ccc} E_i & \xrightarrow{\nu_i} & V_i \\ {\scriptstyle m_e}\downarrow & & \downarrow{\scriptstyle m_v} \\ E_i & \xrightarrow{\nu_i} & V_i \end{array} \tag{25}$$

commutes such that $\nu_i(m_e(E_i)) = m_v(\nu_i(E_i))$. This equivariance constraint yields guidance on what makes for an invalid transform. For example, the single value conversion $\nu_i(x) = .5$ does not commute under translation $t(x) = x + 2$

$$\nu(t(e+2)) \stackrel{?}{=} \nu(e) + \nu(2) \tag{26}$$

$$.5 \neq .5 + .5 \tag{27}$$

On the other hand figure 9 illustrates a valid ν mapping from **Strings** to symbols. The group action on these sets is permutation, so shuffling the words must have an equivalent shuffle of the symbols they are mapped to. To preserve ordinal and partial order monoid actions, ν must be a monotonic function such that given $e_1, e_2 \in E_i$, if $e_1 \leq e_2$ then $\nu(e_1) \leq \nu(e_2)$. For interval scale data, ν is equivariant under translation monoid actions if $\nu(x+c) = \nu(x) + \nu(c)$. For ratio data, there must be equivalent scaling $\nu(xc) = \nu(x)\nu(c)$.

1.3.3 Assembling Marks

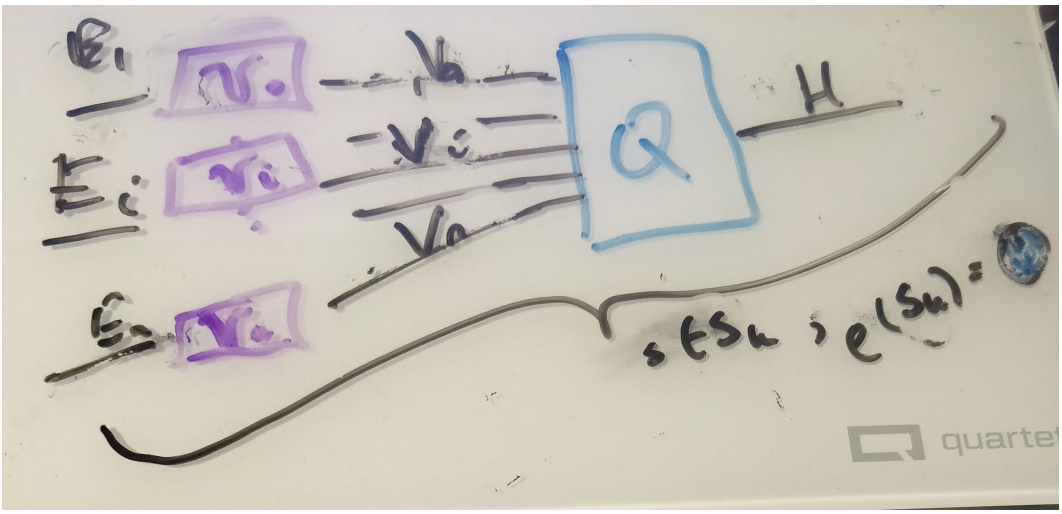
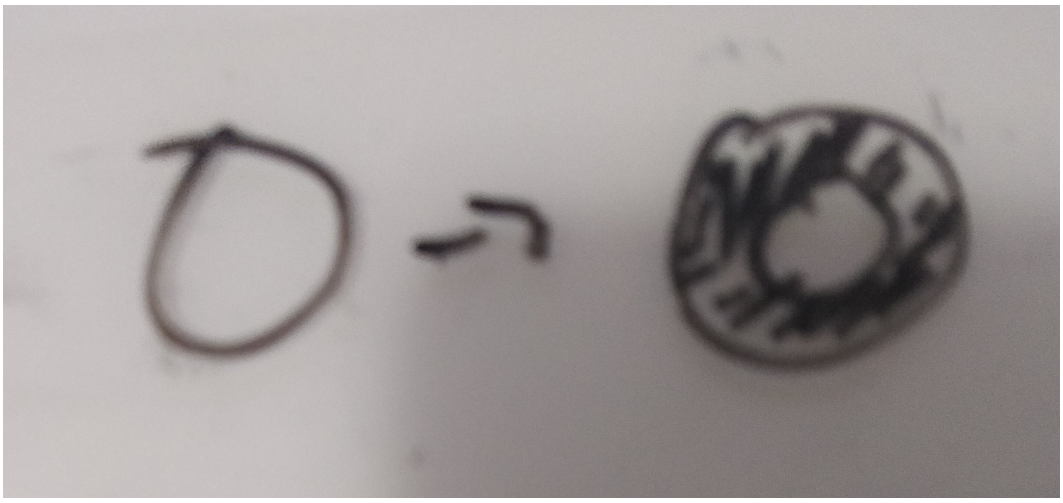


Figure 10: The ν functions convert data E to visual V. Q assembles the different types of visual parameters V_i into a graphic in H. $Q \circ \mu(\xi^{-1}J)$ forms a visual mark by applying Q to a region mapped to connected components $J \subset K$.

As shown in figure **??**, Q takes the individual fields in V as input and outputs a single piece of a graphic on H. As with ν, the constraint on Q is that for every monoid actions on the input in V there is a corresponding monoid action on the output in H

$$Q : \Gamma(V) \rightarrow \Gamma(H) \tag{28}$$

When $Q : \mu \mapsto \rho$ yields a ρ that maps to the same values in D over all S_k, then M can be defined over $\Gamma(H)$ such that a constraint on Q is that it must be equivariant. For example, when μ_i is the color of the glyph, it maps directly to (R,G,B) in D.



Many μ_i are graphical parameters that do not apply to the whole glyph, such as edge thickness in figure **??**.

In these situations, not all ρ in $\Gamma(H)$ will support these parameters; instead we define an action on the output graphic $Q(\Gamma(V)) \in \Gamma(H)$ since by definition every section μ will have a corresponding ρ.

We then define the constraint on Q such that if Q applied to two sections $\mu, \mu\prime$ generate the same graphic ρ, then the output of both sections acted on by the same monoid m must also be the same.

Lets call the visual encodings $\Gamma(V) = X$ and the graphic $Q(\Gamma(V)) = Y$. If $\forall m \in M$ and $\forall \mu, \mu' \in X$,

$$Q(\mu) = Q(\mu') \implies Q(m \circ \mu) = Q(m \circ \mu') \tag{29}$$

then a group action on Y can be defined as $m \circ \rho = \rho'$ where $\rho' = Q(g \circ \mu)$ with $\mu \in Q^{-1}(\rho)$.

Given

- $P = \{xpos, ypos, color, thickness\}$
- $\mu = 0, 0, 0, 1$
- $Q(\mu) = \rho$ generates a piece of the thin circle in figure **??**

the constraint on Q means that the translation $m = \{e, e, e, x + 2\}$ applied to μ such that $\mu' = \{0, 0, 0, 3\}$ has an equivalent action on ρ that causes $Q(\mu\prime)$ to be equivalent to the thicker circle in figure **??**.

Example: Invalid Q Insert some degenerate Q that generates an inconsistent glyph

Check a well defined map $M \times Y \to Y$.

constraint: inputs go to same o utput means changes to inputs mean same changes to output

Graphical Marks To output a mark [3, 6], Q is called with all the regions s that map back to a set of connected components $J \subset K$:

$$J = \{j \in K \text{ s. t. } \exists \gamma \text{ s.t. } \gamma(0) = k \text{ and } \gamma(1) = j\} \tag{30}$$

where the path[7] γ from k to j is a continous function from the interval [0,1].

We define the mark as the graphic generated by $Q(S_j)$

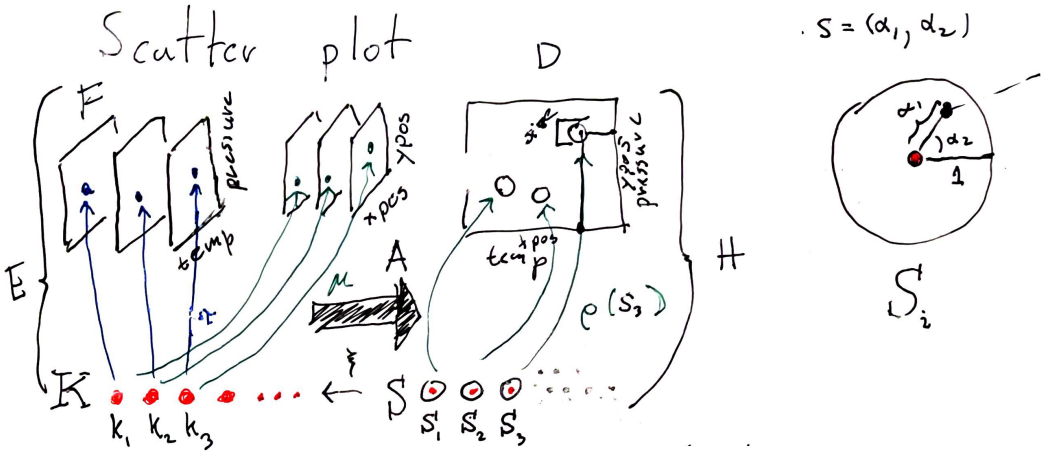
$$H \underset{\rho(S_j)}{\overset{}{\rightleftarrows}} S_j \underset{\xi^{-1}(J)}{\overset{\xi(s)}{\rightleftarrows}} J_k \tag{31}$$

in terms of K because mark is a semantic term denoting the graphic representation of the data.

1.3.4 Sample Qs

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

Q: scatter plot



$$Q(xpos, ypos)(\alpha_1, \alpha_2) \tag{32}$$

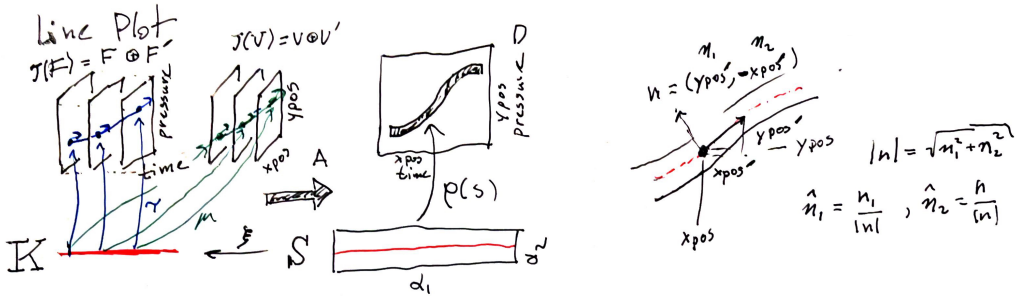
Given a defaut color of black, $\rho_{RGB} = (0, 0, 0)$. The position of this swatch of color can be computed relative to the location on the disc S_i as shown in figure **??**:

$$x = size \bullet \alpha_1 \bullet \cos(\alpha_2) + xpos \tag{33}$$

$$y = size \bullet \alpha_1 \bullet \sin(\alpha_2) + ypos \tag{34}$$

such that $\rho(s) = (x, y, 0, 0, 0)$ where s is the region in H.

Q: line plot



The line plot shown in fig **??** exemplifies the need for the jet discussed in section **??** (needs the normal to push up/down against the normal)

$$Q(xpos, \hat{n_1}, ypos, \hat{n_2})(\alpha_1, \alpha_2) \tag{35}$$

where the magnitude of the thickness is

$$|n| = \sqrt{{n_1}^2 + {n_2}^2} \tag{36}$$

such that the components are

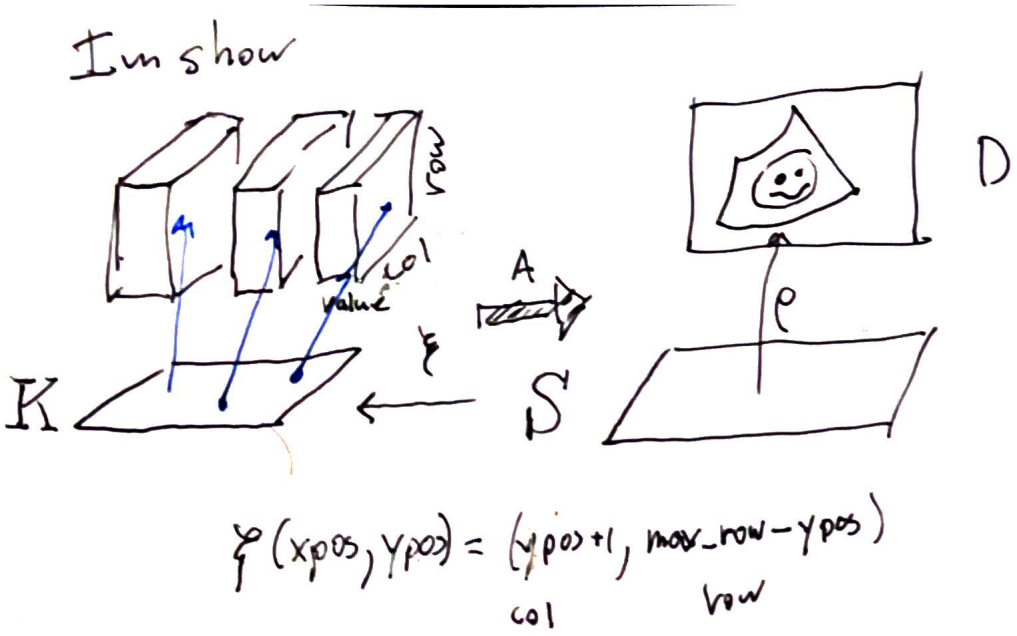
$$\hat{n_1} = \frac{n_1}{|n|}, \ \hat{n_2} = \frac{n_2}{|n|} \tag{37}$$

which yields components of $\rho(s)$:

$$x = xpos(\xi(\alpha_1)) + \alpha_2 \hat{(}n_1)(\xi(\alpha_1)) \tag{38}$$

$$y = ypos(\xi(\alpha_1)) + \alpha_2 \hat{(}n_2)(\xi(\alpha_2)) \tag{39}$$

Q: heatmap



The heatmap in figure **??**

$$Q(xpos, ypos, color) \tag{40}$$

has in the simple case a direct lookup into K to obtain the $\mu = (x, y, c)$ values that are mapped into (arrays are upside down, x is y, is x) - ξ is about translating indices from data to visual alignment

$$D_{RGB} = color(\xi(\alpha_1, \alpha_2))D_x \qquad = xpos(\xi(\alpha_1, \alpha_2))D_y = ypos(\xi(\alpha_1, \alpha_2)) \tag{41}$$

through ρ.

1.4 Making the fiber bundle computable

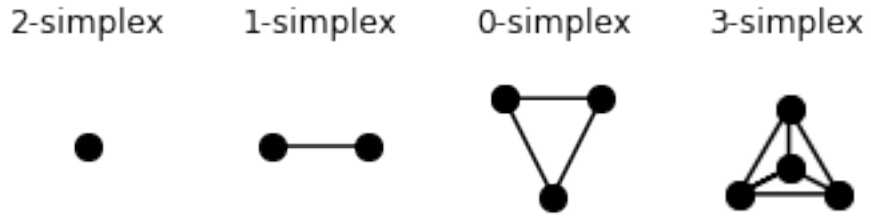


Figure 11: Simplices can encode the connectivity of the data, from fully disconnected (0 simplex) records to all records are connected to at least 3 others

One way to build flexible ξ is to choose a consistent way of representing K. In our draft implementation of the data as fiber bundle model, we triangularize K using complexes of the simplicies shown in figure 11 such that ξ consistently yields some combination of vertexes, edges, and faces. This gives a common data indexing structure on which to build components that could potentially be reused across Q.

By using simplicies, we can then encode the following indexing structure on τ

vertexes $\tau(k)$k= vertex id

edges $\tau(k, \alpha)$, k = edge id, α = distance along edge

faces $\tau(k, \alpha, \beta)$, k = face id, α = x on face, β = y on face

which ξ can use as part of its mapping from visual space back to data space. Path connected components are then sections where $\tau(k_i, 1) = \tau(k_j, 0)$ or the edges of the faces align.

Example: Mobius Strip

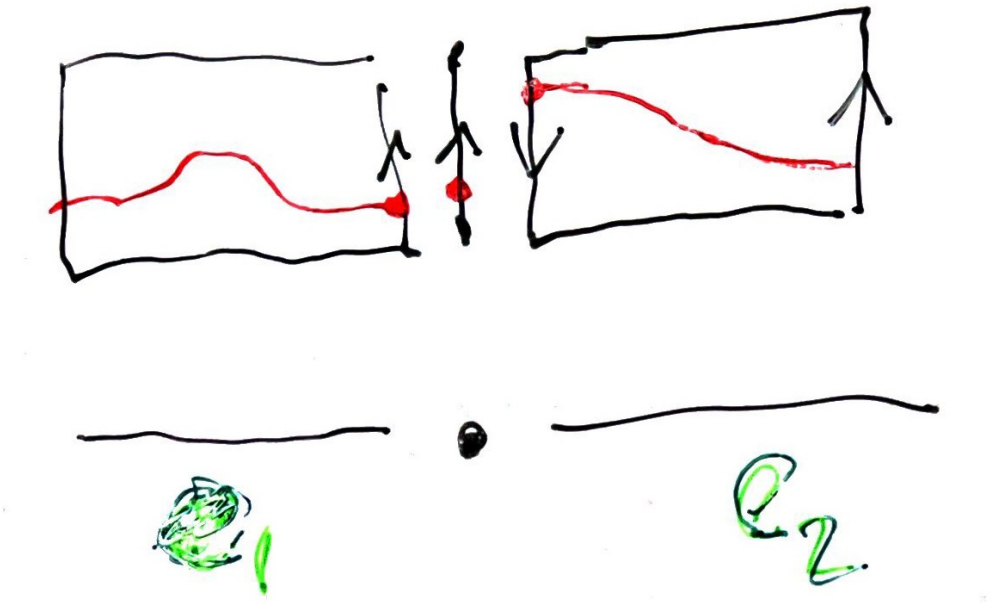


Figure 12: Many non-trivial spaces can be made locally trivial by dividing E into locally trivial subspaces and defining transition functions between the edges on K for how to glue the two subspaces such that the sigmas are continuous.

An example of a non-trivial fiber bundle is a mobius strip[**find the citation for why**];

In figure **??**, the mobius strip is cut into two rectangular base spaces over $K = \{e_1, e_2\}$. We define transition functions from the fiber to the fiber along the edges of the rectangles such that we can glue these rectangles together to reform the mobius strip. A constraint we impose on the transition functions is that the monoid actions are commutative

Example: Torus

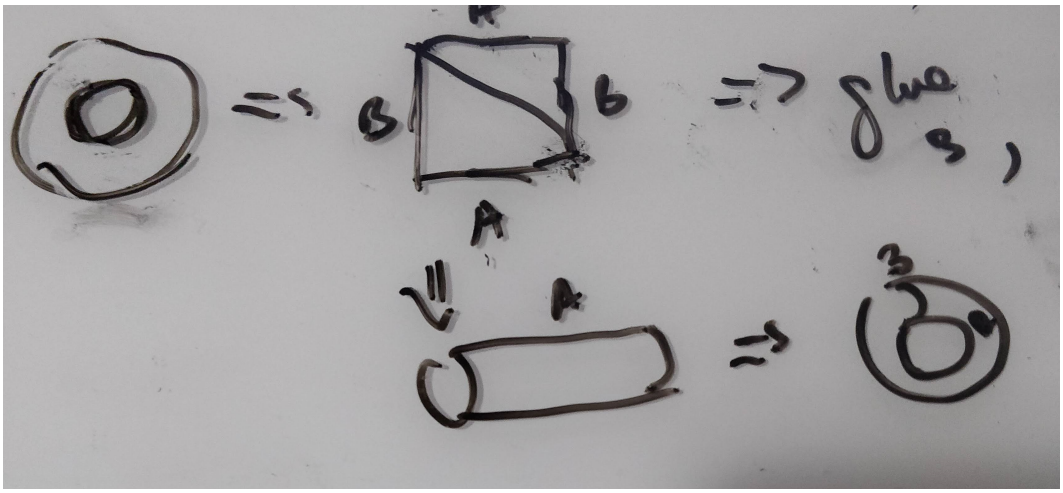


Figure 13: Representation of the base space K of a toridal total space E as two connected 2-simplices.

Given data that lies on the toroidal space shown in figure **??**, the torus E base space K can be implemented as a simplacial complex of two 2-simplexes. We unfold the torus into the two triangles that compose the square; the sections on these triangles are $\tau(triangleidk, \alpha, \beta)$. We also put transition functions on the edges such that A can be glued to A' and B to B' to reconstruct the torus.

1.4.1 Visual Idioms: Equivalance class of artists

As formulated above, every artist function A has fixed ν and generates a distinct graphic ρ. It is unfeasible to implement A for every single graphic; instead we implement the equivalence class of artists $\{A \in A' : A_1 \equiv A_2\}$ which is $Q : \Gamma(V) \to \Gamma(H)$.