

1 Topological Artist Model

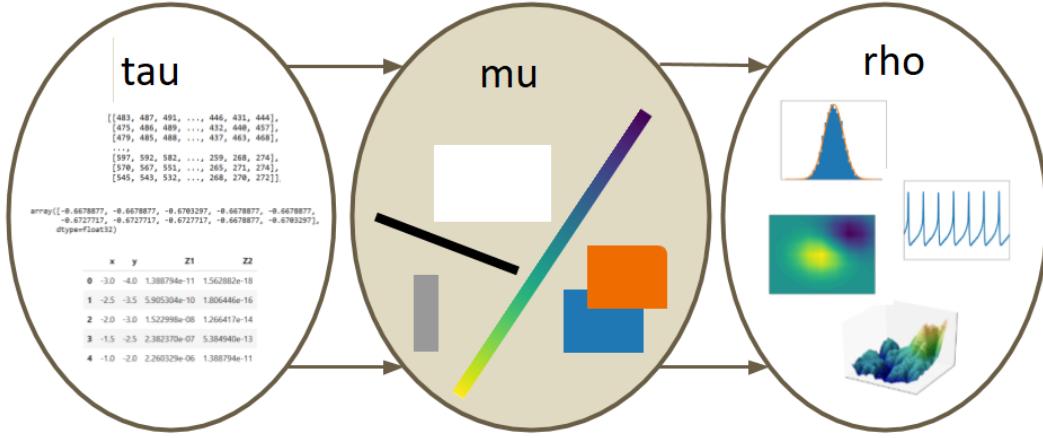


Figure 1: Visualization is equivariant maps between data and visual encoding of the variables and assembly of those encodings into a graphic. *not gonna name these bubbles tau, mu, rho, but might keep the same basic structure of different types of data and encodings*

2 can express continuous functions & not just discrete points Visualization is generally thought
 3 of as structure preserving maps from data into graphics, and in this section we formally define
 4 that structure and how it is preserved via equivariant maps. We can then specify that a
 5 faithful visual mapping is structure preserving, and apply these constraints to visualizations
 6 we may want to develop or implement. We model the data, visual characteristic, and graphic
 7 stages of visualization, shown in figure 1, as topological structures that encapsulate types
 8 of variables and continuity; by doing so we can develop implementations that keep track of
 9 both in ways that let us distribute computation while still allowing assembly and dynamic
 10 update of the graphic.

11 We introduce a mathematical description of the visualization pipeline where artist \mathcal{A}
 12 functions transform data space \mathcal{E} to an intermediate representation in a prerendered graphic
 13 space \mathcal{H} .

$$\mathcal{A} : \mathcal{E} \rightarrow \mathcal{H} \quad (1)$$

14 We first describe how we model data(1.1), graphics(1.2), and intermediate visual char-
 15 acteristics (1.3) as fiber bundles. We then discuss the equivariant maps between data and
 16 visual characteristics (1.3.2) and visual characteristics and graphics (1.3.3) that make up
 17 the artist.

18 1.1 Data Space E

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

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

- ²⁴ F the properties of the variables in the fiber (1.1.1)
²⁵ K the continuity of the records in the base space (1.1.3)
²⁶ τ collections of records (1.1.4).

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

$$F \hookrightarrow E \xrightarrow{\pi} K \quad (2)$$

²⁷ that binds the variables F continuity K . The fiber bundles mentioned in this work
²⁸ are assumed to be trivial[15, 24], unless otherwise specified, because the trivial bundle is
²⁹ $E = K \times F$ such that extra structure in the total space E falls out and discussion can be
³⁰ focused on the fiber and base space.

³¹ 1.1.1 Variables: Fiber Space F

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 [25] as the basis of our fiber space because he binds the components of the fiber to variable names and types. Spivak constructs a set \mathbb{U} that is the disjoint union of all possible objects of types $\{T_0, \dots, T_n\} \in \mathbf{DT}$, where \mathbf{DT} are the data types of the variables in the dataset. He then defines the single variable set \mathbb{U}_σ

$$\begin{array}{ccc} \mathbb{U}_\sigma & \longrightarrow & \mathbb{U} \\ \pi_\sigma \downarrow & & \downarrow \pi \\ C & \xrightarrow[\sigma]{} & \mathbf{DT} \end{array} \quad (3)$$

which is \mathbb{U} restricted to objects of type T bound to variable name c . Given σ , the fiber for a one variable dataset is

$$F = \mathbb{U}_{\sigma(c)} = \mathbb{U}_T \quad (4)$$

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 \mathbb{U}_σ applied to all the columns:

$$F = \mathbb{U}_{T_1} \times \dots \mathbb{U}_{T_i} \dots \times \mathbb{U}_{T_n} \quad (5)$$

which is equivalent to

$$F = F_0 \times \dots \times F_i \times \dots \times F_n \quad (6)$$

- ³² which allows us to decouple F into components F_i .

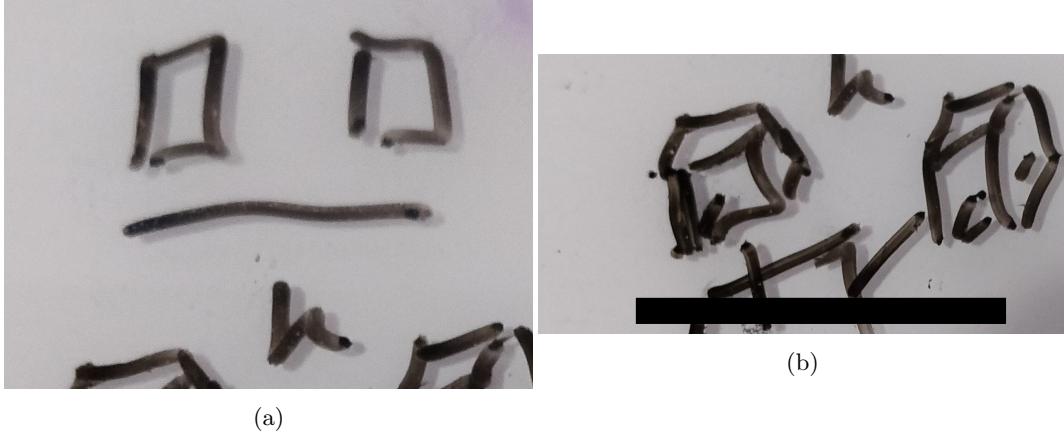


Figure 2: These two datasets have the same base space K but figure 2a has fiber $F = \mathbb{R} \times \mathbb{R}$ which is (time, temperature) while figure 2b has fiber $\mathbb{R}^+ \times \mathbb{R}^2$ which is (time, wind=(speed, direction))

For example, the data in figure 2a is a pair of times and °C temperature measurements taken at those times. Time is a positive number of type `datetime` which can be resolved to positive floats $U_{\text{datetime}} = \mathbb{R}^+$. Temperature values are real numbers $U_{\text{float}} = \mathbb{R}$. The fiber is

$$\mathbb{U} = \mathbb{R}^+ \times \mathbb{R} \quad (7)$$

where the first component F_0 is the set of values specified by ($c_0 = \text{time}$, $T_0 = \text{datetime}$, $\mathbb{U}_\sigma = \mathbb{R}^+$) and F_1 is specified by ($c_1 = \text{temperature}$, $T_1 = \text{float}$, $\mathbb{U}_\sigma = \mathbb{R}$). In figure 2b, 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

$$F = \mathbb{R}^+ \times \mathbb{R}^2 \quad (8)$$

- ³³ such that F_1 is specified by ($c_1 = \text{wind}$, $T_1 = \text{wind}$, $\mathbb{U}_\sigma = \mathbb{R}^2$)

34 1.1.2 Measurement Scales: Monoid Actions

After specifying F we next describe the ways in which we can transform the values by identifying the monoid actions M on the F . We use monoids as the abstraction because they encode compositiblty, which maps well to the data transformation process in a software library [31].

A monoid [17] M_i is a set with an associative binary operator $* : M_i \times M_i \rightarrow M_i$. 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, 23] of M_i is a set F_i with an action $\bullet : M \times F_i \rightarrow F_i$ 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

$$M = M_0 \times \dots \times M_i \times \dots \times M_n \quad (9)$$

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

41 Steven's described the measurement scales[14, 27] in terms of the monoid actions on the
 42 measurements: nominal data is permutable, ordinal data is monotonic, interval data is trans-
 43 latable, and ratio data is scalable [29]. For example, given the fiber ($c = \text{temperature}$, $T =$
 44 float , $\mathbb{U}_\sigma = \mathbb{R}$) which is interval data:

- 45 • monoid operator addition $* = +$
- 46 • monoid operations: $f : x \mapsto x + 1$, $g : x \mapsto x + 2$
- 47 • monoid action operator composition $\bullet = \circ$

then the translation monoid actions on temperature satisfy the condition

$$\begin{array}{ccc} \mathbb{R} & & \\ \downarrow_{x+1^\circ} & \searrow^{(x+1^\circ) \circ (x+2^\circ)} & \\ \mathbb{R} & \xrightarrow{x+2^\circ} & \mathbb{R} \end{array} \quad (10)$$

48 where 1° and 2° are valid distances between two temperatures x .

49 1.1.3 Continuity: Base Space K

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

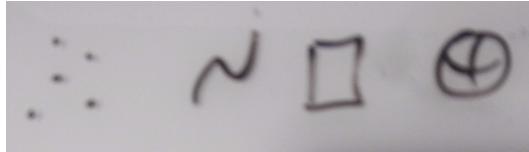


Figure 3: The topological base space K encodes the connectivity of the data space, for example if the data is independent points or a map or on a sphere

54 As illustrated in figure 3, K is akin to an indexing space into E that describes the
 55 structure of E . K can have any number of dimensions and can be continuous or discrete.

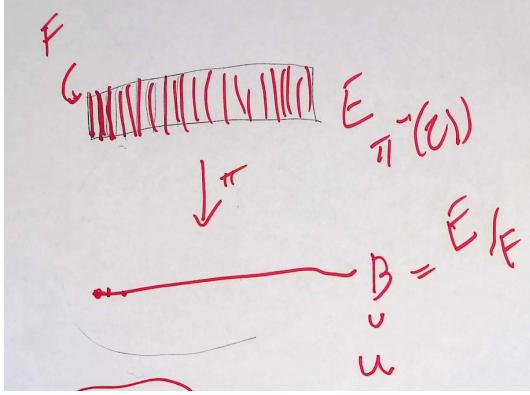


Figure 4: The base space E is divided into fiber segments F . The base space K acts as an index into the records in the fibers. *this figure might be good all the way up top to lay out the components of fb*

56 Formally K is the quotient space [20] of E meaning it is the finest space[2] such that
 57 every $k \in K$ has a corresponding fiber F_k [20]. In figure 4, E is a rectangle divided by
 58 vertical fibers F , so the minimal K for which there is always a mapping $\pi : E \rightarrow K$ is the
 59 line.

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

$$\pi : E_1 \oplus \dots \oplus E_i \oplus \dots \oplus E_n \rightarrow K \quad (11)$$

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

62 The datasets in figure 5 have the same fiber of (temperature, time). In figure 5a the
 63 fibers lie over discrete K such that the records in the datasets in the fiber bundles are
 64 discrete. The same fiber in figure 5b lies over a continuous interval K such that the records
 65 are samples from a continuous function defined on K .

66 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 $\tau : K \rightarrow E$ define the datasets that live in the fiber. We generalize Spivak's description of the section as a table of records [25] to any sort of structured dataset such that

$$\begin{array}{ccc} F & \hookrightarrow & E \\ & \pi \downarrow \nearrow \tau & \\ & K & \end{array} \quad (12)$$

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

$$\tau(k) = (k, (g_{F_0}(k), \dots, g_{F_n}(k))) \quad (13)$$

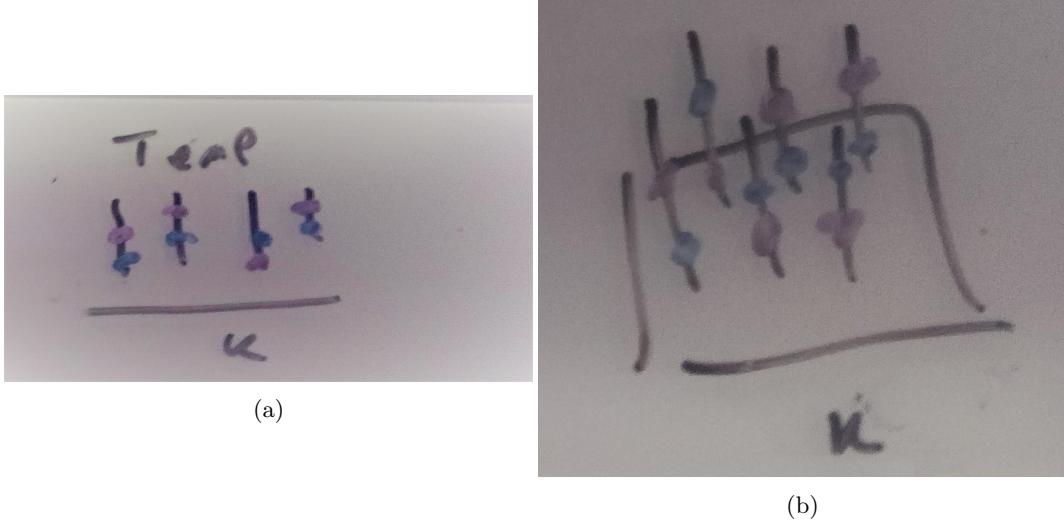


Figure 5: These two datasets have the same (time, temperature) fiber. In figure ?? the total space E is discrete over points $k \in K$, meaning the records in the fiber are also discrete. In figure ?? E lies over the continuous interval K , meaning the records in the fiber are sampled from a continuous space. *revamp figure: F=Plane, k1 = dots, k2=line*

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

$$\tau = (\tau_0, \dots, \tau_i, \dots, \tau_n) \quad (14)$$

⁶⁷ where each section τ_i is a variable or set of variables.

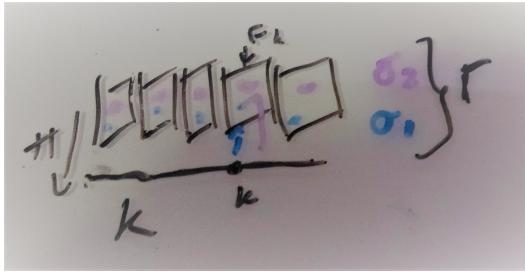


Figure 6: Fiber (time, temperature) with an interval K basespace. The sections τ_i and τ_j are constrained such that the time variable must be monotonic, which means each section is a timeseries of temperature values. They are included in the global set of sections $\tau_1, \tau_2 \in \Gamma(E)$

⁶⁸ In the example in figure 6, the fiber is *(time, temperature)* as described in figure 2
⁶⁹ and the base space is the interval K . The section τ_i resolves to a series of monotonically
⁷⁰ increasing in time records of (time, temperature) values. Section τ_j returns a different
⁷¹ timeseries of (time, temperature) values. Both sections are included in the global set of
⁷² sections $\tau_1, \tau_2 \in \Gamma(E)$.

73 **1.1.5 Sheaf and Stalk**

74 Often a graphic may need to be updated with live data or support zooming in on a segment
 75 of the dataset; to support working with a subset of data, we can use the sheaf $\mathcal{O}(E)$. All fiber
 76 bundles are locally trivial, which means that E restricted over a small enough neighborhood
 77 $U \subset K$ is a locally trivial bundle over U [15]. The sheaf $\mathcal{O}(E)$ is the localized section of
 78 fibers $\iota^*\tau : U \rightarrow \iota^*E$

$$\begin{array}{ccc} \iota^*E & \xleftarrow{\iota^*} & E \\ \pi \downarrow \lrcorner^{\iota^*\tau} & & \pi \downarrow \lrcorner^\tau \\ U & \xleftarrow{\iota} & K \end{array} \quad (15)$$

79 pulled back over the neighborhood U via the inclusion map $\iota : U \rightarrow K$. The localized section
 80 is the germ $\xi^*\tau$. The neighborhood of points k_i surrounding the point k the sheaf lies over
 81 is the stalk \mathcal{F}_b [24, 26]. While E is only the fiber F_k over a specific k , the stalk includes
 82 nearby records because the sheaf lies over the neighborhood U . While this can be useful
 83 for visual transforms, often the extra needed information can be found in the smaller jet
 84 bundle \mathcal{J} [13, 19]. For example, line thickness requires the derivative of the given position
 85 to be rendered, which can be found in $E' = E + \mathcal{J}(E)$

86 **1.2 Graphic: H**

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

91 **1.2.1 Idealized Display D**

The fiber D is an idealized infinite resolution version of the target display space, for example
 a 2D screen or 3D printer. In this work, we assume a 2D opaque image $F = \mathbb{R}^5$ with elements

$$(x, y, r, g, b) \in D \quad (16)$$

92 such that a rendered graphic only consists of 2D position and color. To support overplotting
 93 and transparency, the fiber could be $F = \mathbb{R}^7$ such that $(x, y, z, r, g, b, a) \in D$ specifies the
 94 target display.

95 **1.2.2 Continuity of the Graphic S**

An assumption of graphical representations is that they match the continuity of the data[8,
 28], 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 F . Therefore we define the base space
 mapping from graphic S to data K

$$\begin{array}{ccc} E & & H \\ \pi \downarrow & & \pi \downarrow \\ K & \xleftarrow{\xi} & S \end{array} \quad (17)$$



Figure 7: 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 heatmap has the same continuity in the graphic S as in the data K . **add α, β coordinates to figures**

as the deformation retraction [21] $\xi : S \rightarrow K$ that goes from a region $s \in S_k$ to its associated point s , such that when $\xi(s) = k$, $\xi^*\tau(s) = \tau(s)$. While dimensions can be added to S , it retains the same continuity as K .

In figure 7 each disk S_k indexes how elements in D are glued together to generate a single glyph that is the visual representation of a single record in F_k . For the line, the region β over a point α_i specifies the thickness of the line in S for the corresponding τ on K . The heatmap has the same continuity in data space and graphic space such that no extra dimensions are needed.

1.2.3 Rendering ρ

A section $\rho : S \rightarrow H$ defines a piece of the graphical representation of the data. Evaluated on a single s ρ returns a single element in D . 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 $S_p \subset S$. Integrating over this region on S

$$r_p = \iint_{S_p} \rho_r(s) ds^2 \quad (18)$$

$$g_p = \iint_{S_p} \rho_g(s) ds^2 \quad (19)$$

$$b_p = \iint_{S_p} \rho_b(s) ds^2 \quad (20)$$

yields the color of the pixel p .

As shown in figure 8, the output space queries into the graphic bundle to render the image. We select a pixel p in the output space, inverse map the region of the pixel into $S_p \subset S$, then compute the section ρ over the region S_p . The section 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 $\rho_{rgb}(S_p)$.

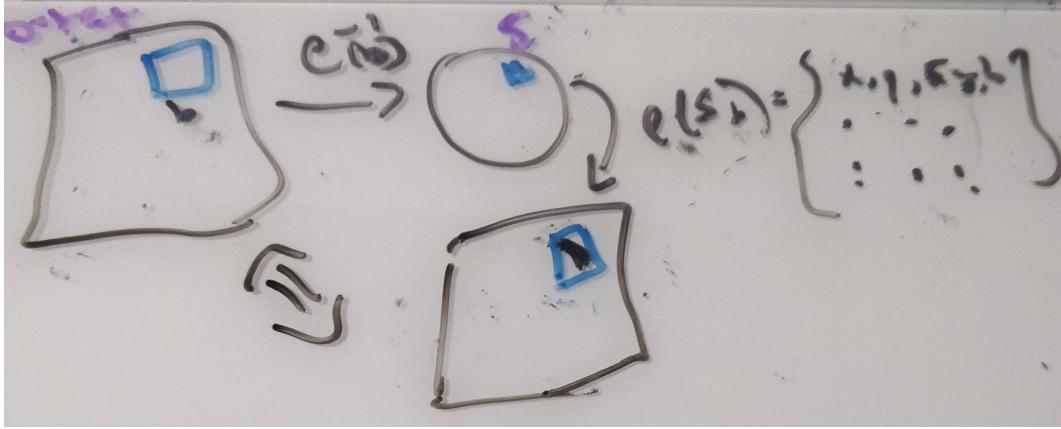


Figure 8: To render a graphic, a pixel p is selected in the display space, which is defined in the same coordinates as the x and y components in D . The inverse mapping $\rho_{xy}(p)$ returns a region $S_p \subset S$. $\rho(S_p)$ returns the list of elements $(x, y, r, g, b) \in D$ that lie over S_p . The integral over the (r, g, b) elements is the color of the pixel.

111 1.3 Artist

112 The artist is the function that converts data into graphics; its name is taken from the
113 analogous part of Matplotlib[12] that builds visual elements to pass off to the renderer. The
114 artist A is a mapping from E padded with data from $\mathcal{J}(E)$ to a graphic that is a section ρ
115 in $\Gamma(H)$

$$\begin{array}{ccccc}
 E' & \xrightarrow{\nu} & V & \xleftarrow{\xi^*} & \xi^*V \xrightarrow{Q} H \\
 & \searrow \pi & \downarrow \pi & \xi^* \pi \downarrow & \swarrow \pi \\
 & & K & \xleftarrow{\xi} & S
 \end{array} \tag{21}$$

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

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

120 of the visual transformation illustrated in figure 1. The functions ξ ν and Q are defined
121 such that they can be evaluated on a single section τ , which allows the artist to be imple-
122 mented such that it does not need all the data. This allows for artists tuned to distributed
123 and streaming data.

124 1.3.1 Visual Fiber Bundle V

125 The visual fiber bundle (V, K, π, P) has section $\mu : V \rightarrow K$ that resolves to a visual
126 variable [3, 18] in fiber P . The fiber space P is defined in terms of the parameters of
127 the visualization specification- for example aesthetics in ggplot [30], channels in vega[22] or
128 parameters in VTK[9] and Matplotlib.

ν_i	μ_i	$\text{codomain}(\nu_i)$
position	x, y, z, theta, r	\mathbb{R}
size	linewidth, markersize	\mathbb{R}^+
shape	markerstyle	$\{f_0, \dots, f_n\}$
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

129 Table 1 is a sample of the fiber space for Matplotlib [11]. A section μ is a tuple of
130 visual values that specifies the visual characteristics of a part of the graphic. For example,
131 given a fiber of $\{xpos, ypos, color\}$ one possible section could be $\{.5, .5, (255, 20, 147)\}$. The
132 $\text{codomain}(\nu_i)$ determines the monoid actions on μ_i . These fiber components are implicit
133 in the library, by making them explicit as components of the fiber we can build consistent
134 definitions and expectations of how these parameters behave.

135 1.3.2 Visual Channels ν

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

$$\{\nu_0, \dots, \nu_n\} : \{\tau_0, \dots, \tau_n\} \mapsto \{\mu_0, \dots, \mu_n\} \quad (22)$$

where $\nu_i : \tau_i \mapsto \mu_i$ is an equivariant map such that there is a monoid homomorphism from F_i to $v\text{fiber}_i$. A validly constructed ν is one where the diagram of the monoid transform m

$$\begin{array}{ccc} E_i & \xrightarrow{\nu_i} & V_i \\ m_x \downarrow & & \downarrow m_v \\ E_i & \xrightarrow{\nu_i} & V_i \end{array} \quad (23)$$

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

$$\nu(t(x + 2)) \stackrel{?}{=} \nu(x) + \nu(2) \quad (24)$$

$$.5 \neq .5 + .5 \quad (25)$$

136 On the other hand figure 9 illustrates a valid ν mapping from **Strings** to symbols. The
137 group action on these sets is permutation, so shuffling the words must have an equivalent

```

[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: In this artis, ν maps the strings to the emojis. For ν to be equivariant, a shuffle in the words should have an equivalent shuffle in the emojis, and a shuffle in the emojis should have an equivalent shuffle in the words.

138 shuffle of the symbols they are mapped to. To preserve ordinal and partial order monoid
 139 actions, ν must be a monotonic function such that given $x_1, x_2 \in E_i$, if $d_{element_1} \leq$
 140 x_2 then $\nu(x_1) \leq \nu(x_2)$. For interval scale data, ν is equivariant under translation monoid
 141 actions if $\nu(x + c) = \nu(x) + \nu(c)$. For ratio data, there must be equivalent scaling $\nu(xc) =$
 142 $\nu(x)\nu(c)$.

143 **1.3.3 Assembling Glyphs Q**

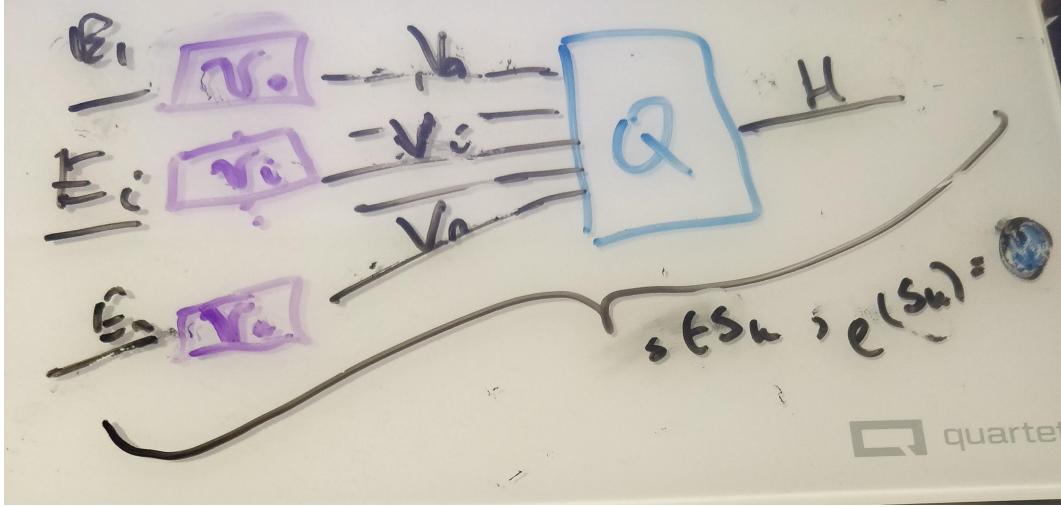


Figure 10: ν functions convert data τ_i to visual characteristics μ_i , then Q assembles μ_i into a graphic ρ such that there is a map ξ preserving the continuity of the data. ρ applied to a region of connected components S_j generates a graphical mark.

144 As shown in figure 10, the assembly function Q combines the fiber F_i wise ν transforms
 145 into a single glyph. Together, ν and Q are a map-reduce operation: map the data into
 146 their visual encodings, reduce the encodings into a glyph. As with ν the constraint on Q is
 147 that for every monoid actions on the input μ there is a corresponding monoid action on the
 148 output ρ .

149 Since we define the equivariant map as $Q : \mu \mapsto \rho$, we define an action on the subset
 150 of graphics $Q(\Gamma(V)) \in \Gamma(H)$ that Q can generate. We then define the constraint on Q such
 151 that if Q is applied to μ, μ' that generate the same ρ then the output of both sections acted
 152 on by the same monoid m must be the same.

Lets call the visual encodings $\Gamma(V) = X$ and the graphic $Q(\Gamma(V)) = Y$. If for all monoids
 $m \in M$ and for all $\mu, \mu' \in X$, the output is equivalent

$$Q(\mu) = Q(\mu') \implies Q(m \circ \mu) = Q(m \circ \mu') \quad (26)$$

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

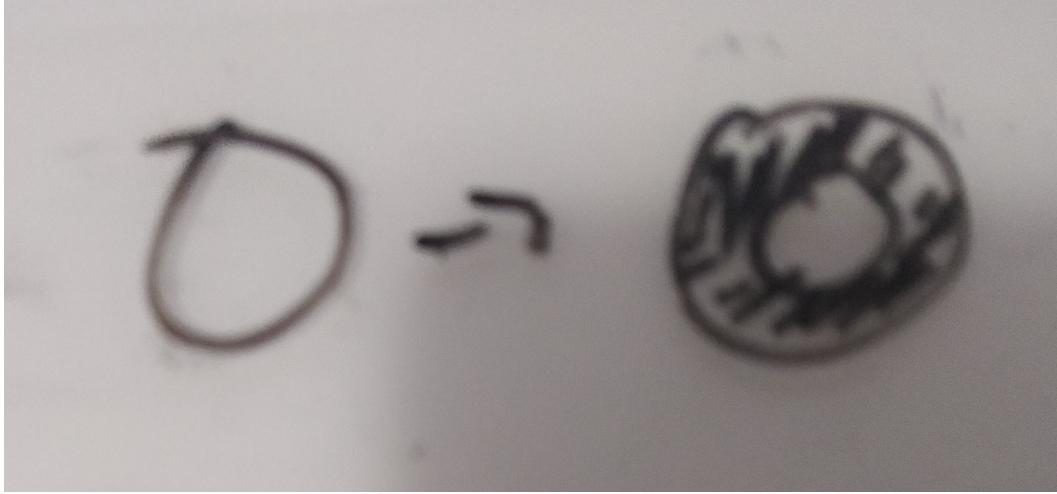


Figure 11: These two glyphs are generated by the same Q function, but differ in the value of the edge thickness parameter μ_i . A valid Q is one where a shift in μ_i is reflected in the glyph generated by ρ .

156 The glyph in figure 11 has the following characteristics P specified by $(xpos, ypos, color, thickness)$
 157 such that one section is $\mu = (0, 0, 0, 1)$ and $Q(\mu) = \rho$ generates a piece of the thin hollow
 158 circle. The equivariance constraint on Q is that the action $m = (e, e, e, x + 2)$, where e is
 159 identity, applied to μ such that $\mu' = (e, e, e, 3)$ has an equivalent action on ρ that causes
 160 $Q(\mu')$ to be equivalent to the thicker circle in figure 11.

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

$$J = \{j \in K \text{ s. t. } \exists \gamma \text{ s.t. } \gamma(0) = k \text{ and } \gamma(1) = j\} \quad (27)$$

161 where the path[7] γ from k to j is a continuous function from the interval $[0, 1]$. We define
 162 the mark as the graphic generated by $Q(S_j)$

$$H \xrightleftharpoons[\rho(S_j)]{} S_j \xrightleftharpoons[\xi^{-1}(J)]{\xi(s)} J_k \quad (28)$$

163 such that for every mark there is at least one corresponding section on K .

164 1.3.4 Sample Glyphs Q

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

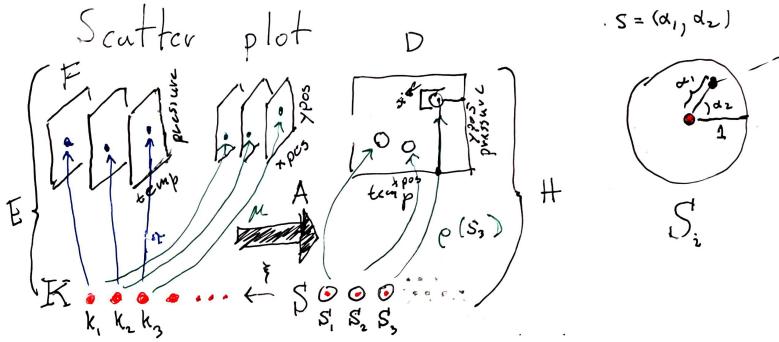


Figure 12: The data is discrete points (temperature, time). Via ν these are converted to (xpos, ypos) and pulled over discrete S . These values are then used to parameterize ρ which returns a color based on the parameters (xpos,ypos) and position α, β on S_k that ρ is evaluated on.

The scatter plot in figure ?? 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 12:

$$x = size \bullet \alpha \bullet \cos(\beta) + xpos \quad (29)$$

$$y = size \bullet \alpha \bullet \sin(\beta) + ypos \quad (30)$$

such that $\rho(s) = (x, y, 0, 0, 0)$ colors the point (x,y) black.

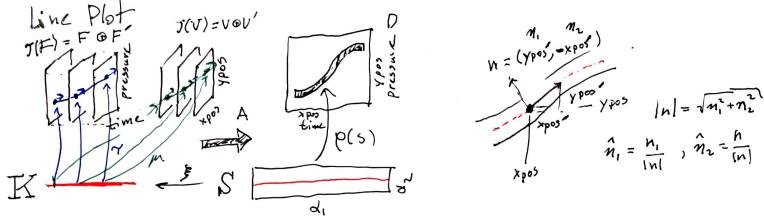


Figure 13: The line fiber (*time, temp*) is thickened with the derivative (*time', temperature'*) because that information will be necessary to figure out the tangent to the point to draw a thick line. This is because the line needs to be pushed perpendicular to the tangent of (xpos, ypos). **this is gonna move once this gets regenerated w/ labels** 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.

The line plot $Q(xpos, \hat{n}_1, ypos, \hat{n}_2)(\alpha, \beta)$ shown in fig 12 exemplifies the need for the jet discussed in section ???. 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 thickness is

$$|n| = \sqrt{n_1^2 + n_2^2} \quad (31)$$

such that the normal is

$$\hat{n}_1 = \frac{n_1}{|n|}, \quad \hat{n}_2 = \frac{n_2}{|n|} \quad (32)$$

which yields components of ρ

$$x = xpos(\xi(\alpha)) + \beta(\hat{n}_1)(\xi(\alpha)) \quad (33)$$

$$y = ypos(\xi(\alpha)) + \beta(\hat{n}_2)(\xi(\alpha)) \quad (34)$$

where (x,y) look up the position $\xi(\alpha)$ on the data and then apply thickness β at that location.

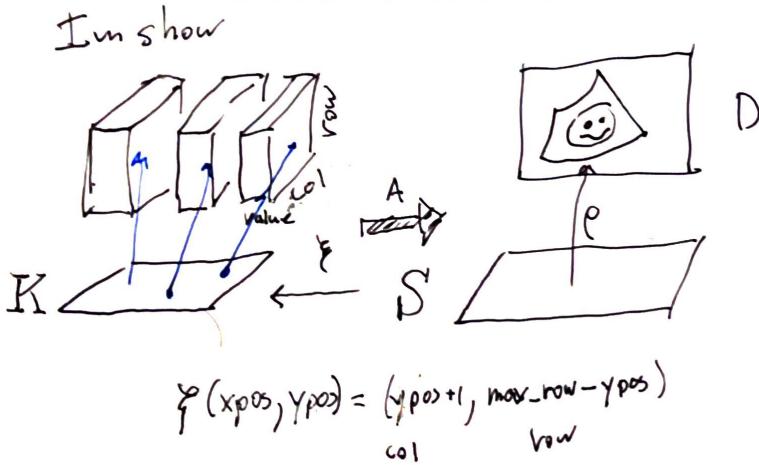


Figure 14: The only visual parameter a heatmap requires is color since ξ encodes the mapping between position in data and position in graphic.

The heatmap $Q(\text{color})$ in figure 14 is a direct lookup $\xi : S \rightarrow K$ such that

$$R = R(\xi(\alpha, \beta)) \quad (35)$$

$$G = G(\xi(\alpha, \beta)) \quad (36)$$

$$B = B(\xi(\alpha, \beta)) \quad (37)$$

where ξ may do some translating to a convention expected by Q for example reorienting the array such that the first row in the data is at the bottom of the graphic.

1.3.5 Assembly factory \hat{Q}

As shown in eq 21, Q is a bundle map $Q : \xi^*V \rightarrow H$ where ξ^*V and H are both bundles over S . Because $\xi^{-1}(k) \subset S$ such that many s go to one k , by definition of the pull back $\xi^*V|_{\xi^{-1}(k)} = \xi^{-1}(k) \times P$. This means that the fiber V is copied over the preimage in $\xi^{-1}(k)$. Given a section $\xi^*\mu$ pulled back from μ on $\pi : V \rightarrow K$ and $s \in \xi^{-1}(k)$, then the section $\xi^*\mu(s) = \xi^*(\mu(k))$ is the image of $(k, \mu(k)) \mapsto (s, \xi^*\mu(s))$ for all s where $\xi(s) = k$. This is illustrated in figure 15, where a μ over J is copied over all $\xi(s) = k$.

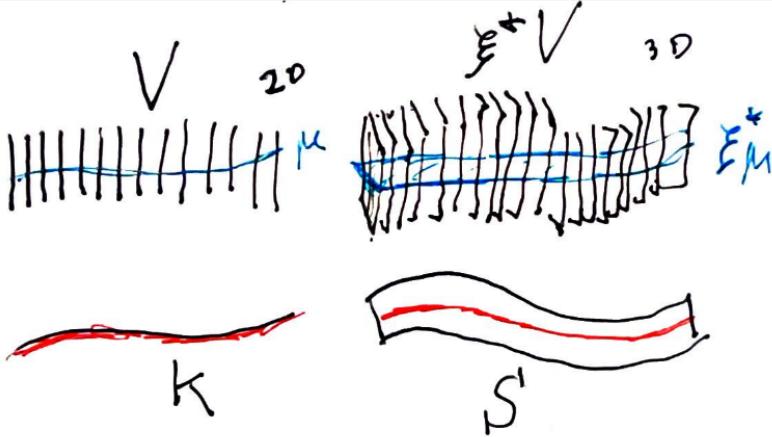


Figure 15: The pullback of the visual bundle ξ^*V is the replication of a μ over all points s that map back to a single k . Because the μ is the same, we can construct a \hat{Q} on μ over k that will fabricate the Q for the equivalent set of s associated to that k

When Q maps sections $Q : \Gamma(\xi^*V) \rightarrow \Gamma(H)$, if we restrict Q input to $\xi^*\mu$, then $Q(\xi^*\mu) = \rho$ depends on s . This means that $\rho(s) := Q(\xi^*\mu)(s)$, but since $\xi^*\mu$ is constant on $\xi^{-1}(k)$, we can define a function $\hat{Q}(\mu(k))(s) := Q((\xi^*\mu)(s))$ where $\xi^{-1}(k) = k$.

In fact, \hat{Q} is a map from visual to graphic $\hat{Q} : \Gamma(V) \rightarrow \Gamma(H)$ locally over k such that $\hat{Q} : \Gamma(V_k) \rightarrow \Gamma(H|_{\xi^{-1}(k)})$. This allows us to construct a \hat{Q} that only depends on K , such that for each $\mu(k)$ there is part of $\rho|_{\xi^{-1}(k)}$. The advantage of \hat{Q} is that S is part of the rendering stage of the pipeline and therefore hidden from the Matplotlib artists and therefore there is no direct way to construct a true Q .

1.3.6 Composition of Artists: +

In this paper we define a simple addition operator that is the disjoint union of fiber bundles E . For example, in figure 16 the scatter plot E_1 and the line plot E_2 have different K that 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 ν position encodings or that the position μ need to be in a specified range. There are situations where $K_2 \hookrightarrow K_1$ that underpin more complex, especially interactive, visualizations; these cases require defining a more complex addition operator that is out of scope for this work.

1.3.7 Equivalence class of artists A'

As formulated above, every artist function A has fixed ν and Q which generates a distinct graphic ρ . It is impractical to implement an artist for every single graphic; instead we implement the equivalence class of artists $\{A \in A' : A_1 \equiv A_2\}$. Equivalent artists have the same fiber bundle V and same assembly function Q but act on different sections μ .

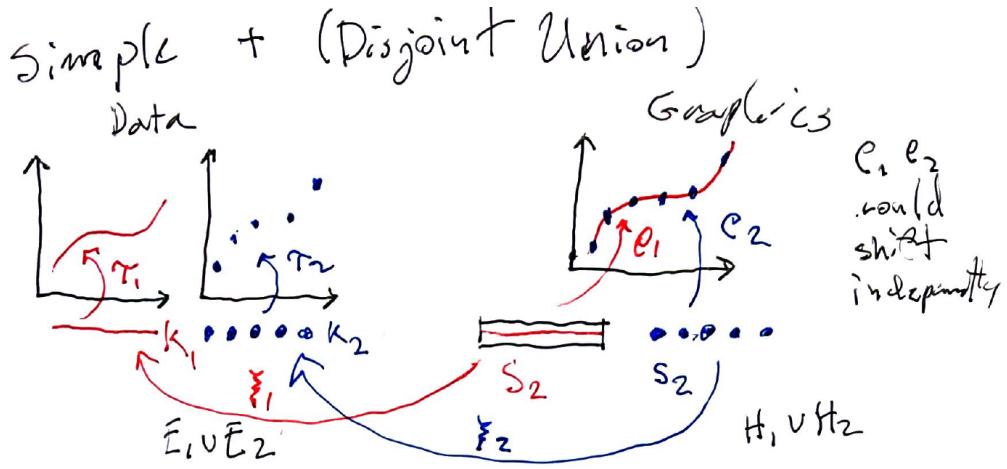


Figure 16: τ_1 and τ_2 are distinct datasets passed through artists A_1 and A_2 to generate graphics ρ_1 and ρ_2 . These graphics happen to be rendered to the same image, but otherwise have no intrinsic link.

204 To further simplify implementation, we identify a minimal P associated with each A' that
 205 defines what visual characteristics of the graphic must originate in the data such that the
 206 graphic is identifiable as a given chart type.

Figure 17: Each of these graphics is generated by a different artist A which is the equivalence class of scatter plots A'
this is gonna be a whole bunch of scatter plots

207 For example, a scatter plot of red circles is the output of one artist, a scatter plot of
 208 green squares the output of another, as are the rest of the graphs in figure ???. These two
 209 artists are equivalent since their only difference is in the literal visual encodings (color,
 210 shape). Shape and color could also be defined in Q but the position must come from the
 211 fiber $P = (xpos, ypos)$ since fundamentally a scatter plot is the plotting of one position
 212 against another[8]. We also use this criteria to identify derivative types, for example the
 213 bubble chart[28] is a type of scatter where by definition the glyph size is mapped from the
 214 data.

215 1.4 Making the fiber bundle computable

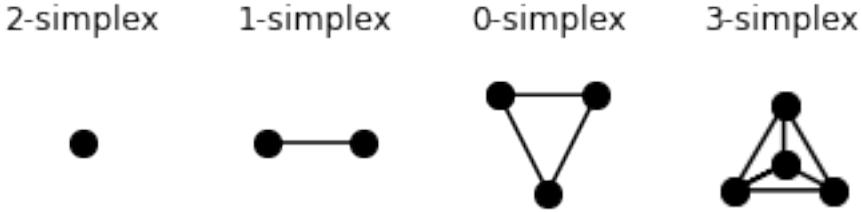


Figure 18: Simplices can encode the connectivity of the data, from fully disconnected (0 simplex) records to all records are connected to at least 3 others

216 One way of expressing the connectivity of records in a dataset is to implement K as a
217 simplicial complex, which is a set of simplices such as those shown in figure 18. The
218 advantage of triangulation is that it is general enough to work for more complex topology
219 based visualization methods [10] while also providing a consistent interface of vertices, edges,
220 and faces for ξ to map into. When triangulated, the simplices encode the continuity in the
221 data

simplex	continuity	τ
vertex	discrete	$\tau(k)$
edge	1D	$\tau(k, \alpha)$
face	2D	$\tau(k, \alpha, \beta)$

Table 2

222 such that each section is bound to a simplex $k \in K$. As shown in table 2, in a 1D
223 continuous spaces each τ lies distance α along edge k , while in a 2D continuous space each
224 τ lies at coordinate α, β on the face k . This is directly analogous to indexing to express
225 connectivity in N-D arrays, while also natively supporting graphs and trees as they are
226 simplicial complexes of nodes and edges. Path connected components are then sections
227 where edges or faces meet.

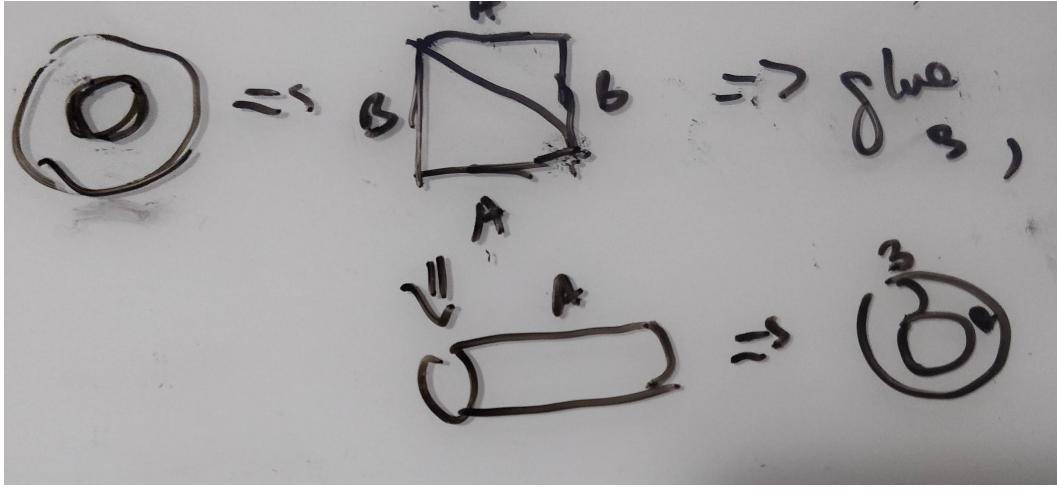


Figure 19: The torus E is unraveled into a simplicial complex of 2 faces K . Transition functions are defined on the edges of K such that surface can be glued back into the torus.
add cross sections a and b to ring and color same as edges in complex

228 One way of encoding the torus in figure 19 while retaining the continuity of both cross
 229 sections a, b is to unravel it into a simplicial complex of two triangles with labeled edges.
 230 Transition functions δ are defined on the edges of K such that a can be glued to a' and b to b' to
 231 reconstruct the torus. This simplicial complex is then used as the base space encoding the
 232 continuity of data that lies in the torus. A constraint on the transition functions is that the
 233 monoid actions on the fibers on the edges of E are commutative $M * F \mapsto \delta(MF) = M * \delta(F)$

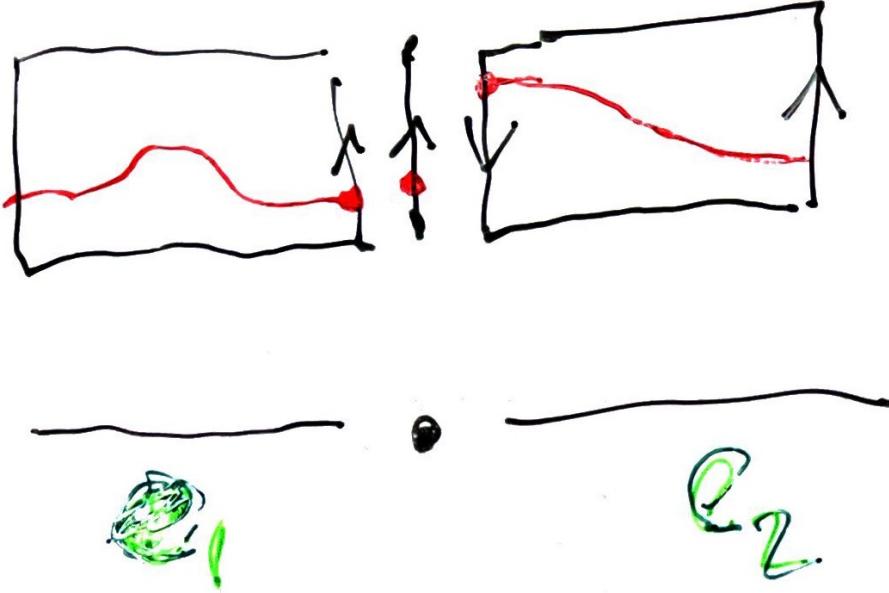


Figure 20: Many non-trivial spaces can be made locally trivial by dividing E into locally trivial subspaces and defining transition functions between the edges on K for how to glue the two subspaces such that the τ are continuous.

234 Another advantages of triangulization is that it provides a way to encode non-trivial
 235 structures such as the mobius strip[16]. As shown in figure 20, one way of making the
 236 mobius strip trivial is to seperate it into two spaces E_1 and E_2 and then define transition
 237 functions that specify that the edges of E_1 need to be reversed to line up with E_2 such that
 238 the sections along the edges meet. As with the torus, the transition functions must preserve
 239 monoid commutativity.