Make any stupid plot you want

Hannah Aizenman

A dissertation proposal submitted to
the Graduate Faculty in Computer Science in partial fulfillment of the
requirements for the degree of Doctor of Philosophy,
The City University of New York

Committee Members:
Dr. Michael Grossberg (Advisor), Dr. Robert Haralick, Dr. Lev Manovich,
Dr. Huy Vo

Abstract

Contents

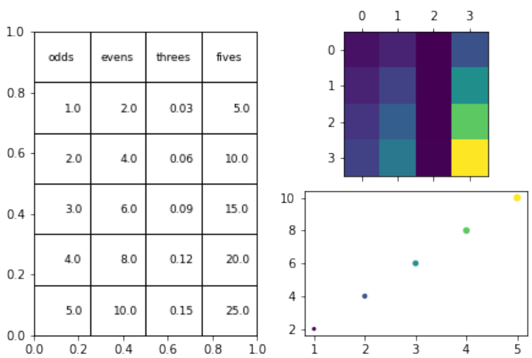


Figure 1: Implicit in visualization is the assumption that these three representations of data are equivalent, specifically that the measurements within a variable and relations of the measurements of each variable are preserved.

1 Introduction

1.1 Thesis statement

We define a visualization as a transform from data to graphic that preserves the topology of the data and faithfully map the properties of the measurement type. In fig 1, we implicitly assume that the translation from table to heatmap has preserved the order of observations (the rows) and that the perceptually uniform sequential colormap has been applied such that the ordering relation on floats matches the ordering on the colormap (darker colors map to larger numbers). We also make this assumption about color in the scatter map, and that the translation to size and position on screen also respect the ordering on floats. In this work, we propose to mathematically describe the transform of data to visual space such that we can make explicit the implicit topology and types visualizations preserve. We then propose a new architecture for the Python visualization library Matplotlib [8] based on these descriptions because the Matplolib artist layer is analogous to the transforms.

1.2 What is a viz

? Acquired codes of meaning

2 Not all data are tables

Tables, images (Lev), graphs (network X)

set up: dubois

theorists: bertin, munzner, mackinlay

talk about: matplotlib arch paper, excel/matlab arch, vtk & ggplot (compare/contrast, we're blending these things)

3 Notation & Definitions

In this section we introduce a mathematical description of the visualization pipeline where artist $\mathfrak{A}$ functions transform data space $\mathfrak{E}$ to an intermediate representation in a prerendered display space $\mathfrak{H}$.

$$\mathcal{A} : \mathcal{E} \rightarrow \mathcal{H} \tag{1}$$

We use fiber bundles[7, 19] to model data and graphics because they allow to seperate concerns

- E is a locally trivial fiber bundle over K representing data space.
- H is a fiber bundle over S representing visual space
- K and S are a triangulizable topological space or a CW complex encoding the connectivity of points in E and H respectively

The fiber bundles mentioned in this work are assumed to be locally trivial[11, 21].

3.1 Data Space E

As proposed by Butler [3, 4], we model data as a fiber bundle (E, K, π, F)

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

with topological total space E, base space K, fiber space F, and the map from total space to base space $\pi : E \rightarrow K$. Maps from K to E are called sections and select specific points in K. The global space of sections in E is $\Gamma(E)$.

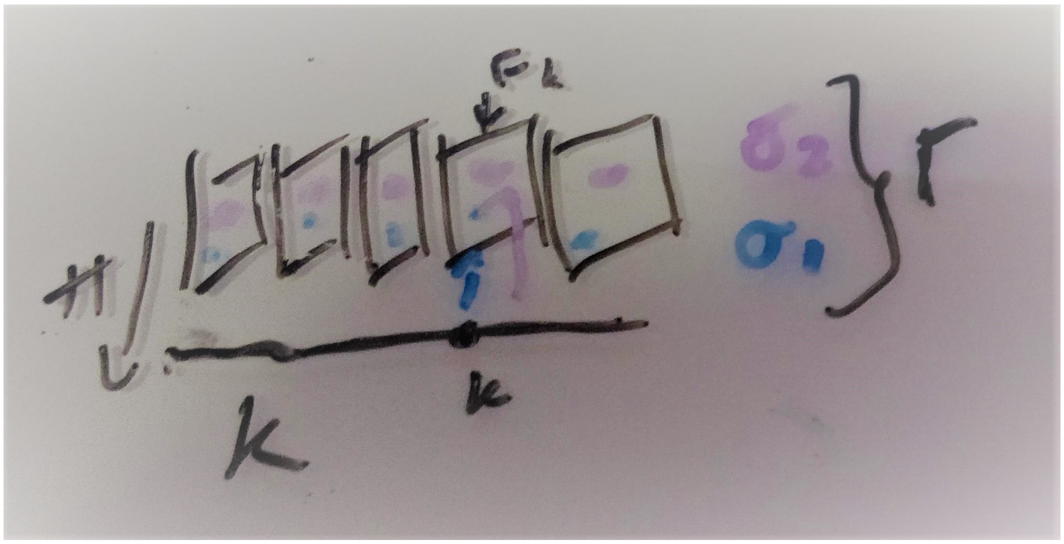


Figure 2: F is an $n \times m$ space of points. The section τ_1 returns the blue points, while τ_2 returns the purple points. $\Gamma(E)$ is the set of all sections, including τ_1 and τ_2

.

Example The fiber bundle in figure **??** can encode different types of data. F is a cartesian product of the codomains of 2 variables and K is a line on the interval [0,1]. The sections τ_1 and τ_2 are different sets of data values in the space.

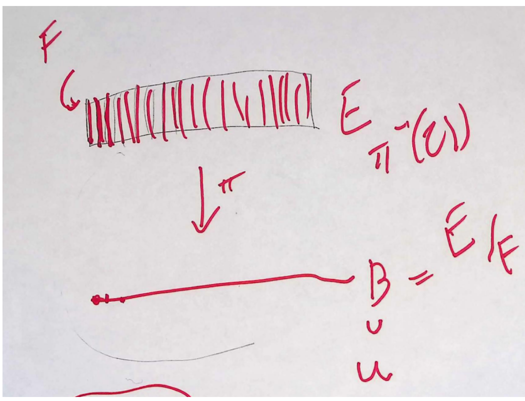
distribution F is the cartesian cross product of temperature values and the range of probabilities [0,1] such that every point in E is a (temperature, probabilty) pair. τ_1 and τ_2 each return a different distribution of temperatures.

timeseries $F = \{\text{temperature}\} \times \{\text{timestamps}\}$, such that every point is a (temperature, time) pair. τ_1 and τ_2 each return different timeseries of temperature.

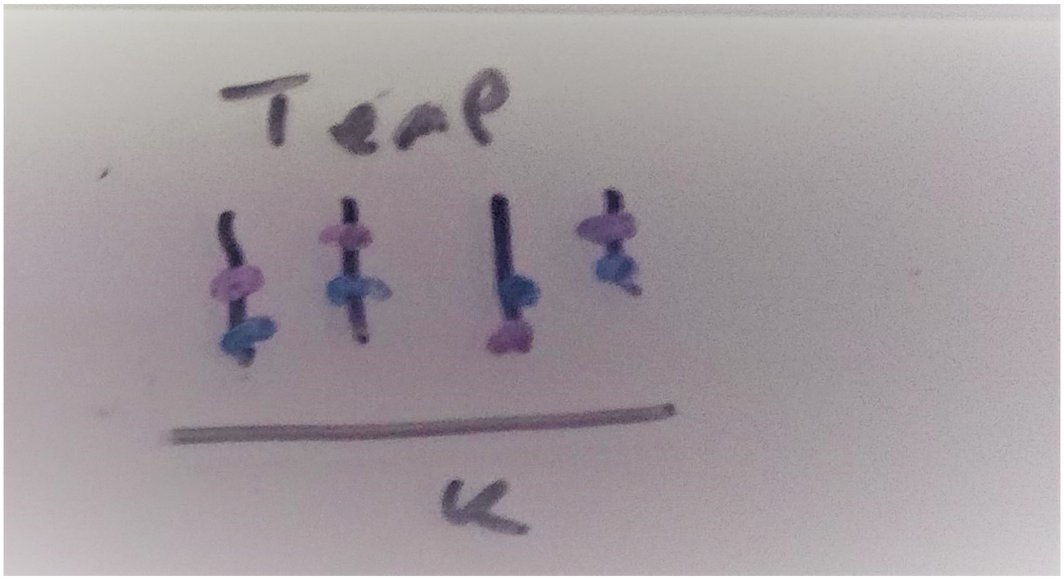
3.1.1 Base Space $\mathcal{K}$

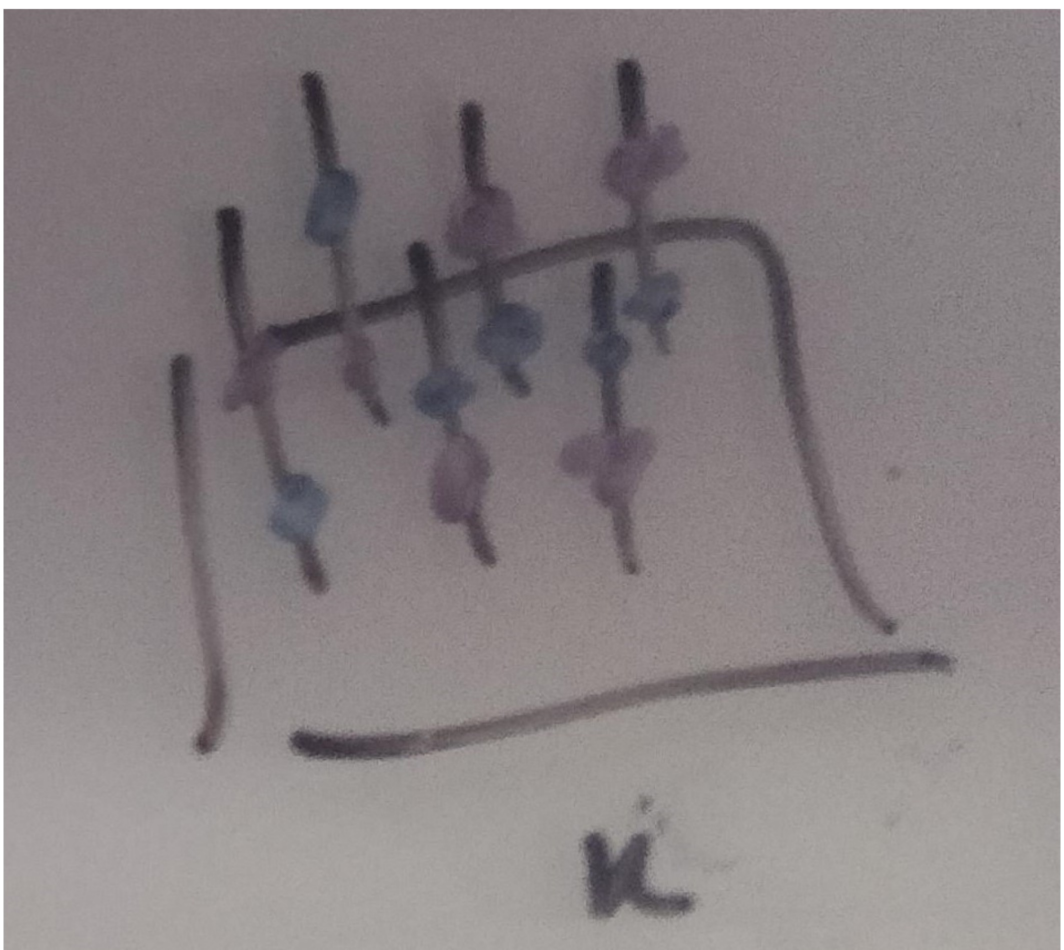
Figure 3: The topological base space K encodes the connectivity of the data space, for example if the data is independent points or a map or on a sphere

Data can be discrete observations, timeseries, maps, fields [15] and K is a set of points k that can act as keys from a representative space, such as seen in figure **??**, to the data values in E [15].



K and F are not intrinsic to E, rather they are choices in how E is subdivided[17]. In figure **??** we can divide a rectangular base space such that there is a short fiber and long base space or a long fiber and short base space. This is analogous to long and wide forms of the same table [24].


Temp
k


k

Example in figure **??**, temperature is the only one data field in r but the K base spaces are different. subfig[1] is a timeseries, so the temperature in r at time t is dependent on the temperature in r_{t-1} and the temperature in r_{t+1} is dependent on r_t; this connectivity is expressed as a one dimensional K where K is the number line. In the case of the map, every temperature in r is dependent on its nearest neighbors on the plane, and one way to express this is by encoding K as a plane. K does not know the time or latitude or longitude of the point as those are metadata variables describing the k rather than the value of k. The mapping $\tau : K \to E$ provides the binding between the key $k \in K$ and the value r in E [14].

Triangulization The base space K is a representation of the connectivity of the data, specifically whether the points in E are discrete or sampled from a continous space. The same dataset can be expressed with different K.

In our draft implementation of the data as fiber bundle model, we represent K as a simplacial complex.

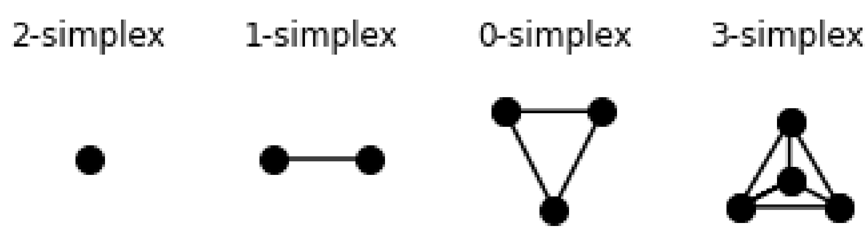


Figure 4: Simplices encode the connectivity of the data, from fully disconnected (0 simplex) observations to all observations are connected to at least 3 other observations. Higher order simplicies are outside the scope of this paper.

K is a triangularizable topological space; one triangularization scheme is as a set composed of simplices, such as those shown in figure **??**.

Example chopping up a torus maybe? talk about how that gets unpacked into triangles and then into vertices

3.1.2 Fiber Space

$$\begin{tikzcd} \pi^{-1}(U) \arrow[r, "\varphi"] \arrow[d, "\pi"'] & U \times F \arrow[dl, "\mathrm{proj}_U"] \\ U & \end{tikzcd} \tag{3}$$

such that $\varphi : \pi^{-1}(U) \to U \times F$ is a homeomorphism where π and proj_U both map to U and the fiber over k

$F_k = \pi^{-1}(k)$ is homomorphic to the fiber F.

Definition A monoid[12] M is a set that is closed under an associative binary operator $*$ and has an identity element $e \in M$ such that $e * a = a * e = a$ for all $a \in M$. A left monoid action [1, 20] of M is a set X with an action $\bullet$ with the properties:

closure $\bullet : M \times X \to X$,

associativity for all $m, t \in M$ and $x \in X, m \bullet (t \bullet x) = (m \bullet t) \bullet x$

identity for all $x \in X, e \in M, e \bullet x = x$

$M = M_1 \times M_2 \times \ldots \times M_n$

Example

3.1.3 Section

The section τ is the mapping from base space to total space $\tau : K \to E$

$$\begin{tikzcd} F \arrow[r, hook] & E \arrow[d, "\pi"'] \\ & B \arrow[u, bend right, "f"'] \end{tikzcd} \tag{4}$$

such that f is the right inverse of π

$$\pi(f(k)) = k \text{ for all } k \in K \tag{5}$$

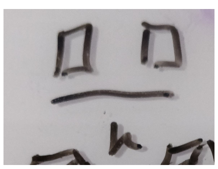
In a trivial fiber bundle, $E = K \times F$ [7, 19]:

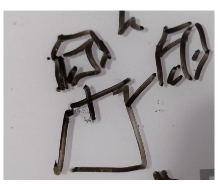
$$f(b) = (b, g(b)) \tag{6}$$

where the domain of $g(b)$ is F_b and returns a point p in F_b. The space of all possible sections f of E is $\Gamma(E)$. All sections $f \in \Gamma(E)$ have the same fibers F and connectivity K.

Example For each field $c \in C$, the record function $r : C \to U_\sigma$ returns an object of type $\sigma(c) \in DT$. The set of all records $\Gamma(\sigma)$ is the set of all sections on U_σ. Spivak defines the τ mapping from an index of databases K to records $\Gamma(\sigma)$ as $\tau : K \to \Gamma(\sigma)$. This is equivalent to $\tau : k \to E$ since $F = \Gamma(\sigma)$ and F is the embedding in E on which the records r lie.

3.1.4 Example





The fiber in figure **??** is the space of possible temperature values in degrees celsius, such that $F = [temp_{min}, temp_{max}]$ and is named Temp. In figure **??** time is encoded as a second dimension. This means that the set of possible values F with $C = \{\text{Temp}, \text{Time}\}$:

$$F = [temp_{min}, temp_{max}] \times [time_{min}, time_{max}] \tag{7}$$

and the function τ that retrieves records from F is

$$\tau(k) = (k, (r : \text{Temp} \rightarrow temp, r : \text{Time} \rightarrow time)) \tag{8}$$

$$temp \in [temp_{min}, temp_{max}], time \in [time_{min}, time_{max}]) \tag{9}$$

Since $\tau(k) = (k, r)$, *temp* is bound to a named data field and *sigma* binds *temp* to a temperature data type.

3.1.5 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 $\mathcal{O}(E)$:

$$\begin{tikzcd}
\iota^* E \arrow[d, shift right, "\pi"'] & E \arrow[l, "\iota^*"'] \arrow[d, shift right, "\pi"'] \\
U \arrow[u, shift right, bend right, "\iota^*\tau"'] & K \arrow[l, "\iota"] \arrow[u, shift right, bend right, "\tau"']
\end{tikzcd} \tag{10}$$

As shown in equation 3, there is a local space $U \subset K$ around every k. The inclusion map $\iota : U \rightarrow K$ is pulled back such that $\iota^* E$ is the space of E restricted over K. The localized section of fibers $\iota^*\tau : U \rightarrow \iota^* E$ is the sheaf with a germ of $\xi^*\tau$. The neighborhood of points k_i surrounding the point k the sheaf lies over is the stalk $\mathcal{F}_b$ [21, 22].

The jet bundle $\mathcal{J}$ [9, 16] is a type of sheaf that allows for writing differential equations on sections of fiber bundles; this information is required for some visual characteristics, such as line thickness.

3.2 Prerender Space H

We define a graphic space H such that we do not have to assume the physical output space of the renderer. This means that the graphic in H can be output to a screen or 3D printed space or a dome.

We model the prerender space as a fiber bundle (H, S, π, D). H is the predisplay space, with a fiber D dependent on the target display and a base space of S.

3.2.1 Base space

The underlying topology S of a graphic often needs more dimensions than the data topology K because of the specifications of the display space. For example, a line plot on a plane (such as a screen or a piece of paper) by necessity needs to also have a thickness so that it is visible, which maps back to a set of connected points in H. The topology of these connected points is therefore the region $s \subset S$ such that $\xi : S \to K$ is a deformation retraction [18]

$$\begin{tikzcd} E \arrow[d, "\pi"'] & H \arrow[d, "\pi"'] \\ K & S \arrow[l, "\xi"'] \end{tikzcd} \tag{11}$$

that goes from a region $s \in S_k$ to its associated point k, such that when $\xi(s) = k$, $\xi^* \tau(s) = \tau(k)$.

3.2.2 Fiber and Section

A section $\rho : S \to H$ is a mapping from a region s on a mathematical encoding of the image to a region xy on the screen that the renderer then maps to visual space as defined in D.

Example For a physical screen display, we can consider a predisplay space that is a trivial fiber bundle $H = \mathbb{R}^5 \times S$ such that ρ is

$$\rho(s) = [x(s), y(s), r(s), g(s), b(s)] \tag{12}$$

To draw an image, a region, H is inverse mapped into a region $s \in S$ where

$$s = \rho^{-1}_{XY}(xy) \tag{13}$$

such that the rest of the fields in $\mathbb{R}^7$ are then integrated over s to yield the remaining fields:

$$r = \iint_s \rho_R(s) ds^2 \tag{14}$$

$$g = \iint_s \rho_G(s) ds^2 \tag{15}$$

$$b = \iint_s \rho_B(s) ds^2 \tag{16}$$

Here we assume a single opaque 2D image such that the z and $alpha$ fields can be omitted. To support overplotting and transparency, we can consider $D = R^7$

3.2.3 Example

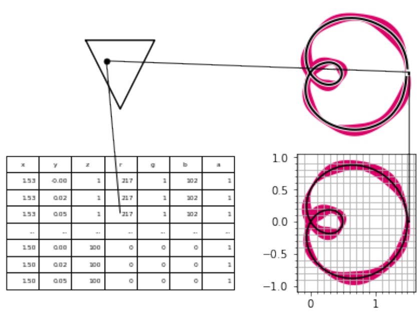


Figure 5

As illustrated in figure 5, words.

3.3 Artist

In this section we will define the artist as a mapping from a sheaf $\mathcal{O}(E)$ to $\mathcal{O}(H)$.

$$A : \mathcal{O}(E) \rightarrow \mathcal{O}(H) \tag{17}$$

The artist decomposes to mapping data to visual $\nu : E \rightarrow V$, then compositing V pulled back along ξ to ξ^*V to a visual mark in prerender space $Q : \xi^*V \rightarrow H$.

$$
\begin{tikzcd}
E \arrow[r, "\nu"] \arrow[dr, "\pi"'] & V \arrow[d, "\pi"] & \xi^*V \arrow[l, "\xi^*"'] \arrow[r, "Q"] \arrow[d, "\xi^*\pi"'] & H \arrow[dl, "\pi"] \\
 & K & S \arrow[l, "\xi"] &
\end{tikzcd} \tag{18}
$$

The pullback map ξ^* copies each value in V over k to s in corresponding S_k such that ξ^*V can have multiple values that map to one value in V.

The visual fiber bundle $(V,\ K,\ \pi,\ P)$ has section $\mu : V \to K$ that resolves to a visual variable [2, 13] in fiber P. The visual transformer ν is a set of functions each targeting a different μ

$$\{\nu_0, \ldots, \nu_n\} : \{\tau_0, \ldots, \tau_n\} \mapsto \{\mu_0, \ldots, \mu_n\} \tag{19}$$

where μ_i are the visual parameters in the assembly function $Q(\mu_0, \ldots, \mu_n)(s) = \rho(s)$.

3.3.1 Example: Matplotlib Visual Fiber

For example, for Matplotlib [8], some of the possible types in P are: Table **??** is a

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

3.3.2 Visual Channels

$\nu : E \to V$ is an equivariant map such that there is a homomorphism from left monoid actions on E_i to left monoid actions on V_i where i identifies a field in the fiber. E_i and V_i each contain a set of values as defined in F and P respectively. A validly constructed ν is one where the diagram

$$\begin{tikzcd} E_i \arrow[r, "\nu_i"] \arrow[d, "m_e"'] & V_i \arrow[d, "m_v"] \\ E_i \arrow[r, "\nu_i"] & V_i \end{tikzcd} \tag{20}$$

commutes such that $\nu_i(m_e(E_i)) = m_v(\nu_i(E_i))$.

Example: Ordering To preserve ordering of elements in E_i, ν must be a monotonic function such that given $e_1, e_2 \in E_i$

$$\text{if } e_1 \leq e_2 \text{ then } \nu(e_1) \leq \nu(e_2) \tag{21}$$

Example: Translation According to Stevens, interval data is a set with general linear group actions [10, 23]. Position is a visual variable that can support translation

$$\nu(x + c) = \nu(x) + \nu(c) \tag{22}$$

Example: Invalid ν Given a transform $t(x) = x + 2$, we construct a ν that always takes data to .5:

$$\begin{tikzcd} E_1 \arrow[r, "\lambda : e \mapsto .5"] \arrow[d, "2e"'] & V_i \arrow[d, "2v"] \\ E_1 \arrow[r, "\lambda"] & V_1 \end{tikzcd} \tag{23}$$

This ν is invalid because the graph does not commute for t:

$$\nu(t(e)) \stackrel{?}{=} t(\nu(e)) \tag{24}$$

$$.5 \stackrel{?}{=} t(.5) \tag{25}$$

$$.5 \neq 2 * .5 \tag{26}$$

To consutruct a valid ν, the diagram must commute for all monoid actions on the sets in E_i, V_i.

3.3.3 Assembling Marks

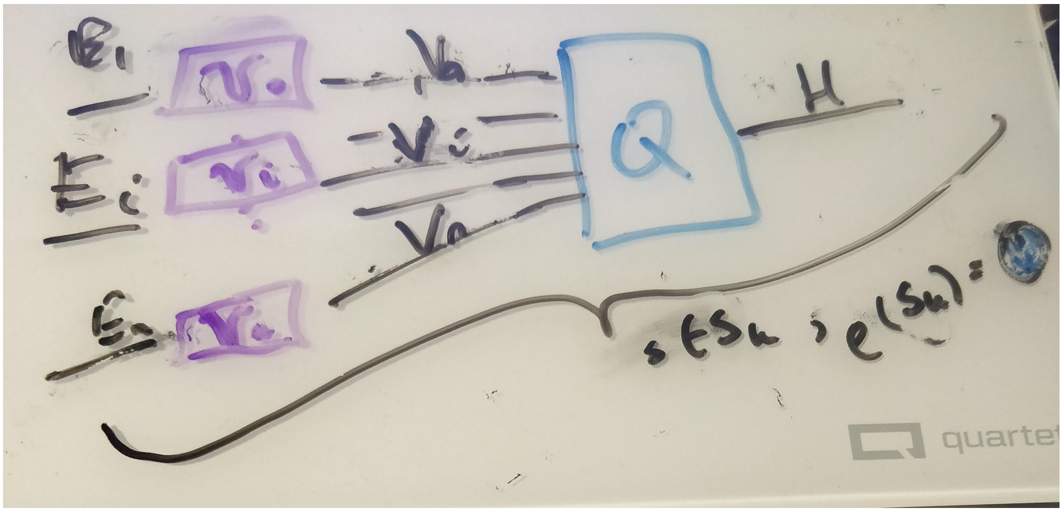


Figure 6: The ν functions convert data E to visual V. Q assembles the different types of visual parameters V_i into a graphic in H. $Q \circ \mu(\xi^{-1}J)$ forms a visual mark by applying Q to a region mapped to connected components $J \subset K$.

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:

$$Q : \Gamma(V) \to \Gamma(H) \tag{27}$$

$Q : \mu \mapsto \rho$

We want a monoid/group action on $Q(\Gamma(V))$, do not need an action on all of $\Gamma(H)$

To output a mark [2, 5], Q is called with all the regions s that map back to a set of connected components $J \subset K$:

$$J = \{j \in K \text{ s. t. } \exists \gamma \text{ s.t. } \gamma(0) = k \text{ and } \gamma(1) = j\} \tag{28}$$

where the path[6] γ from k to j is a continous function from the interval [0,1].

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

$$H \underset{\rho(S_j)}{\overset{}{\leftrightarrows}} S_j \underset{\xi^{-1}(J)}{\overset{\xi(s)}{\leftrightarrows}} J_k \tag{29}$$

where the set $j \subset J$ is the set of marks in the graphic.

$M = M_1 \times M_2 \times \ldots \times M_n$ color X xpos (Cross product of monoid actions over V/E), cartesian product b/c independent. M over E was translated to acting over $\Gamma(V)$ of these act on $\Gamma(V)$ ν translates the sets, constraint on ν is equivariance of M. Is limited to the $Q(\Gamma(V)) \subset of\Gamma(H)$ because not all visual tramsforms (μ) are supported by all ρ. So we define the actions M on the image of Q, $Q(\Gamma(V)) = Y$. We can backward define our actions?

When the visual atrtribute in μ is some kind of direct property of D, then we can define M on $\Gamma(H)$ and require that Q preserves it. But we have graphical parameters that do not apply to the whole glyph and therefore are not direct mappings on D, such as facecolor or line thickness. For these types of properties, we need to define an action on the target graphic $Q(\Gamma(V)) \in \Gamma(H)$.

If can check this on Q, then you can define an equivariant M. Use action of M on V to define action M on Y.

Let ....

If $\forall g \in M$ and $\forall \mu, \mu\prime \in X$,

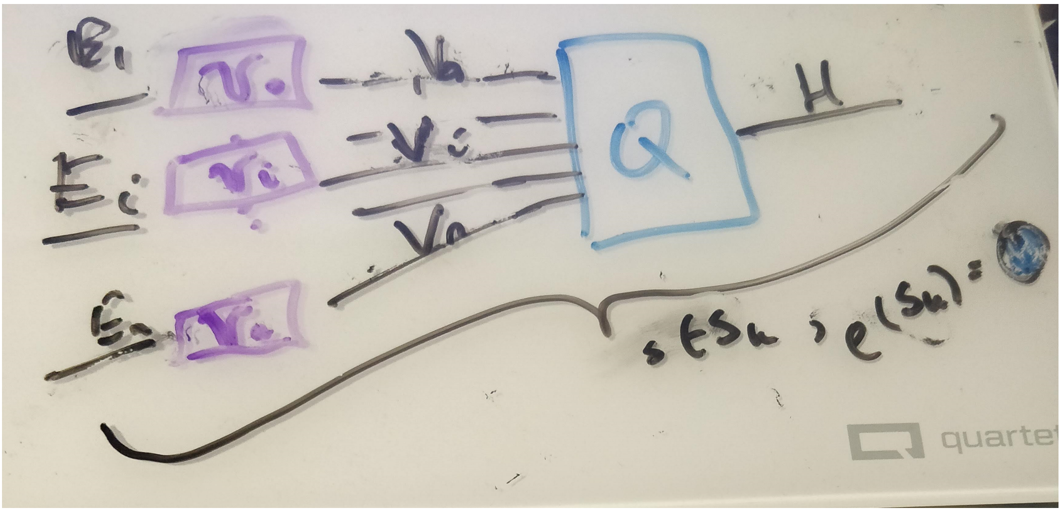
$$Q(\mu) = Q(x2) \tag{30}$$

If true, then we can define a group action on Y is defined as $g \circ \rho = \rho\prime$ where $\rho\prime = Q(g \circ \mu)$ with $\mu \in Q^{-1}(\rho)$.

tuple of figure above Is ... if two μ go to the same graphic(glyph), then if we transform them the same way they need to generate the same graphic in H. two sections of V if they map to the same thing, they need to map to the same thing if you transform they need to map to the same thing.

change g to m

Example: Invalid Q Check a well defined map $M \times Y \to Y$.



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

3.3.4 Visual Idioms: Equivalance class of artists

Given $O(E)$ of the same type that output to the same type of graphic $O(H)$, the

Natural transformation + composition is partial ordering? Back and forth is equivalent

4 Matplottoy

We build on top of the existing Matplotlib architecture [8] so that we can initially focus on the data to graphic transformations and rely on Matplotlib for the other graphical elements of the visualization and the rendering. For a primitive graph such as a bar, scatter, line, our path is: python

E = Section() V = 'parameter': ('field': transformer)

fig, ax = plt.subplots() artist = Q(E, V) ax.add$_{a}rtist(artist)$

4.1 Data: $\mathcal{E}$

We propose a semantic markup of the fiber and connectivity that we will later use to check the constraints of the ν and Q. python class FiberBundle: def $_{init}{}_{(self,K,F):"""K:'tables':[]F:'fieldname':'type':datatype,'monoid':meas}$

def is$_section(self, section) : """checksifasectionisfromagivenfiberbundle : arevaluesinF, arekeysinK"""pass$

We add a monoid field to the schema like structure of the fiber since the monoid actions are the constraints on the ν and Q. These will usually be identified by a measurment scale such as 'nominal', 'ordinal', 'interval', and 'ratio'. The optional? range field can be used to specify restrictions on the type such as set of string categorical values or fixed interval of float values. We implement K using the triangleization scheme described in section 3.1.1. This means we expect data to be provided as tables where the name encodes the connectivity:

vertex - disconnected points

edge - 1D continuity along the edge

face - 2D continuity along the face

Nested continuity is encoded in the fiber. For example, one way to encode a movie is a 1D timeseries of Frame objects, where each Frame is a nm continuous array.

4.1.1 Sections

As described in section 3.1.3, the data values are encode as the section of the fiber bundle. We implement this as as a wrapper class around a data container object of the form: python class Section: FB = FiberBundle(K, F)

def $_init_{(self,*args):passdefview(self,simplex="vertex"):convertdataintoatomiccolumnorderreturnfieldname:data}$

The view function is the closest analog to τ, as it returns a a data container type object. We specify field based top level indexing so that we can pair the fields with transformer ν objects.

4.2 Encoding: $\mathcal{V}$

We implement ν as encoder objects with an optional '$_init_{'methodthatcanbeusedtospecifytherangeoftargetvisualvariables.}$

python class Encoder: def $_init_{(self,args):passdefvalidate(self,monoid):checkiftransformsupportsmonoidactionpassdefconvert(self,value):con}$

The validate method specifies the monoids for which this a valid ν and checks against the specification in the fiber. The convert method converts a value from data space to an internalized normal form as described in table **??**.

4.3 Graphic: $\mathcal{H}$

We inherent from the current Matplotlib artists for our Q, which is responsible for constructing a curried fully specified internal representation of the graphic. python class Q(BaseClass): required = optional = def $_{init_{(self,data,*args,kwargs):checkthatrequiredandoptionalvisualparametersarepassedincheckthatth}}$ While equation 18 specifies that the values of μ can be generated before they are input to Q, we curry the opertaions of obtaining the data and performing the transforms to the draw method because that allows us to more easily propogate updates to the data to the screen.

4.4 Case Study: Iris(Penguins)

References

[1] *Action in nLab*. https://ncatlab.org/nlab/show/action#actions_of_a_monoid.

[2] Jacques Bertin. “II. The Properties of the Graphic System”. English. In: *Semiology of Graphics*. Redlands, Calif.: ESRI Press, 2011. ISBN: 978-1-58948-261-6 1-58948-261-1.

[3] D. M. Butler and M. H. Pendley. “A Visualization Model Based on the Mathematics of Fiber Bundles”. en. In: *Computers in Physics* 3.5 (1989), p. 45. ISSN: 08941866. DOI: 10.1063/1.168345.

[4] David M. Butler and Steve Bryson. “Vector-Bundle Classes Form Powerful Tool for Scientific Visualization”. en. In: *Computers in Physics* 6.6 (1992), p. 576. ISSN: 08941866. DOI: 10.1063/1.4823118.

[5] Sheelagh Carpendale. *Visual Representation from Semiology of Graphics by J. Bertin*. en.

[6] “Connected Space”. en. In: *Wikipedia* (Dec. 2020).

[7] "Fiber Bundle". en. In: *Wikipedia* (May 2020).

[8] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science Engineering* 9.3 (May 2007), pp. 90–95. ISSN: 1558-366X. DOI: 10.1109/MCSE.2007.55.

[9] "Jet Bundle". en. In: *Wikipedia* (Dec. 2020).

[10] W A Lea. "A Formalization of Measurement Scale Forms". en. In: (), p. 44.

[11] *Locally Trivial Fibre Bundle - Encyclopedia of Mathematics*. https://encyclopediaofmath.org/wiki/Locally_trivial

[12] "Monoid". en. In: *Wikipedia* (Jan. 2021).

[13] T Munzner. "Marks and Channels". In: *Visualization Analysis and Design*, pp. 94–114.

[14] Tamara Munzner. "Ch 2: Data Abstraction". In: *CPSC547: Information Visualization, Fall 2015-2016* ().

[15] Tamara Munzner. "What: Data Abstraction". In: *Visualization Analysis and Design*. AK Peters Visualization Series. A K Peters/CRC Press, Oct. 2014, pp. 20–40. ISBN: 978-1-4665-0891-0. DOI: 10.1201/b17511-3.

[16] Jana Musilová and Stanislav Hronek. "The Calculus of Variations on Jet Bundles as a Universal Approach for a Variational Formulation of Fundamental Physical Theories". In: *Communications in Mathematics* 24.2 (Dec. 2016), pp. 173–193. ISSN: 2336-1298. DOI: 10.1515/cm-2016-0012.

[17] "Quotient Space (Topology)". en. In: *Wikipedia* (Nov. 2020).

[18] "Retraction (Topology)". en. In: *Wikipedia* (July 2020).

[19] Todd Rowland. *Fiber Bundle*. en. https://mathworld.wolfram.com/FiberBundle.html. Text.

[20] "Semigroup Action". en. In: *Wikipedia* (Jan. 2021).

[21] E.H. Spanier. *Algebraic Topology*. McGraw-Hill Series in Higher Mathematics. Springer, 1989. ISBN: 978-0-387-94426-5.

[22] "Stalk (Sheaf)". en. In: *Wikipedia* (Oct. 2019).

[23] S. S. Stevens. "On the Theory of Scales of Measurement". In: *Science* 103.2684 (1946), pp. 677–680. ISSN: 00368075, 10959203.

[24] Hadley Wickham et al. "Tidy Data". In: *Journal of Statistical Software* 59.10 (2014), pp. 1–23.