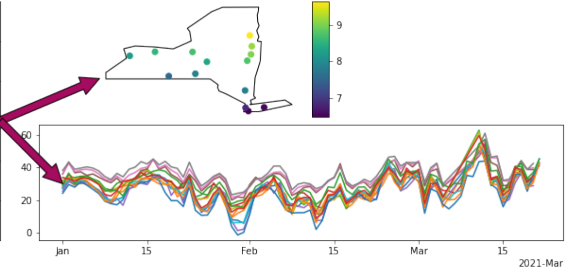
Topological Equivariant Artist Model

March 26, 2021

Hannah Aizenman
Advisor: Dr. Michael Grossberg
Committee: Dr. Robert Haralick, Dr. Lev Manovich, Dr. Huy Vo
External Member: Dr. Marcus Hanwell

Visualizations are structure preserving maps

DATE	LATITUDE	LONGITUDE	TAVG
2021-01-01	42.1	-77.1	27.86
2021-01-01	40.6386	-73.7622	35.96
2021-01-01	42.7	-75.5	30.02
2021-01-01	43.0078	-73.6511	27.68
2021-01-01	41.5	-73.9	29.48



equivariance properties of data and visual encoding match

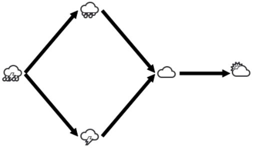
continuity connectivity of data and visual encoding match

Domain specific libraries assume data structure[HA06]

DATE	LATITUDE	LONGITUDE	TAVG
2021-01-01	42.1	-77.1	27.86
2021-01-01	40.6386	-73.7622	35.96
2021-01-01	42.7	-75.5	30.02
2021-01-01	43.0078	-73.6511	27.68
2021-01-01	41.5	-73.9	29.48

ggplot[Wic16]
Vega[SWH14]
Altair[Van+18]
Tableau [STH02]
[Han06; MHS07]

ImageJ[SRE12]
ImagePlot[Stu21]
Napari[Sof+21]



Gephi[BHJ09]
Graphviz[Ell+02]
Networkx[HSS08]

General purpose libraries can't[TM04]

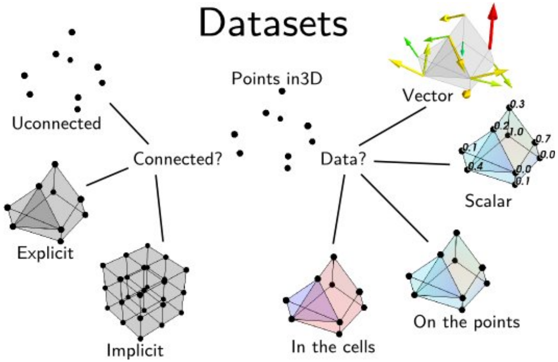


Figure: Data Representation, MayaVi 4.7.2 docs[Dat]

1. Matplotlib[Hun07] →Seaborn[Wt20], xarray [HH17]
2. D3 [BOH11]
3. VTK [Han+15; Gev+12](MayaVi[RV11])→ Titan[BJ09], ParaView[AGL05]

Best practices in visualization design

Expressiveness structure preserving mappings[Mac86]

Graphical Integrity graphs show **only** the data[Tuf01]

Naturalness easier to understand when properties match[Nor93]

Contributions

Topological	continuity
Equivariant	monoid action
Artist	Matplotlib $artist : data \to graphic$
Model	

Topological Equivariant Artist Model

An artist $\mathscr{A}$ is an equivariant map

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

from data $\mathscr{E}$ space to graphic $\mathscr{H}$ space.

Model data as a fiber bundle [BB92; BP89]

A fiber bundle is a tuple (E, K, π, F) defined by the map π

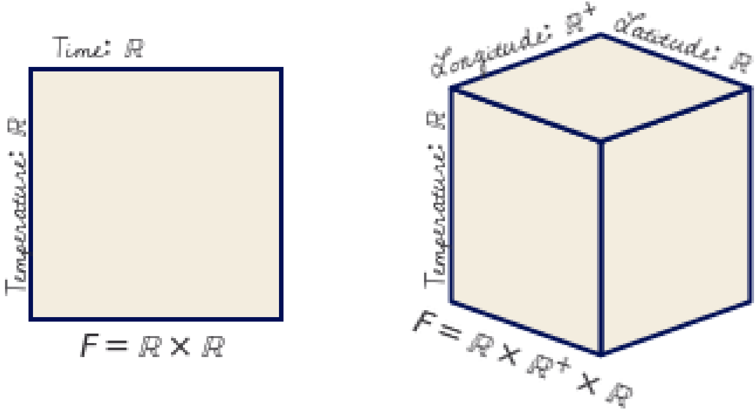
$$F \hookrightarrow E \xrightarrow{\pi} K$$

total space E topology

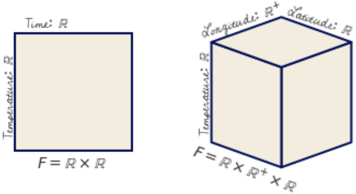
fiber space F schema

base space K continuity

Encode variable types in a schema like fiber [Spi10; Spi]



Monoids are the structure of the components of F



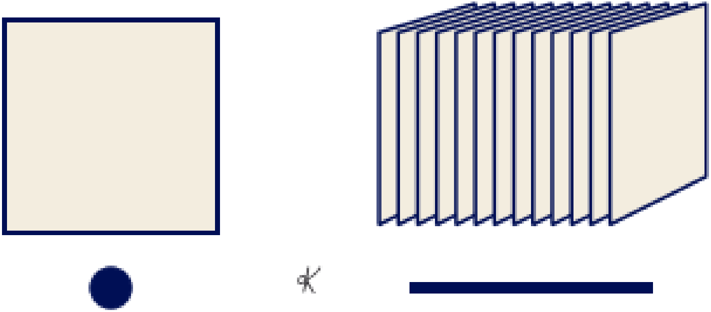
$$F = F_0 \times \ldots \times F_i \times \ldots \times F_n$$

Monoid actions M_i (e.g. rotation, partial ordering) define the structure on F_i

$$\bullet : M_i \times F_i \rightarrow F_i$$

where $\bullet$ is associative and has an identity action.

K is an indexing space



K is the space of *keys* into data in E[Mun14]

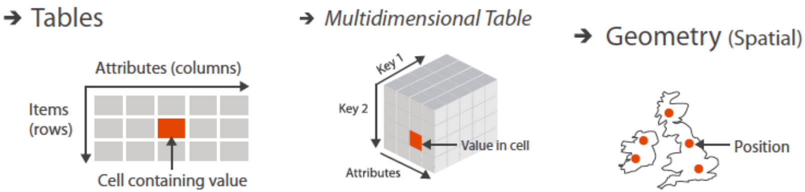
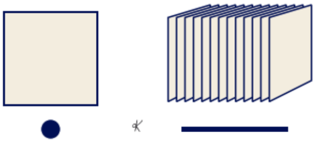


Figure: Figure 2.8 in Munzner's Visualization Analysis and Design[Mun14]

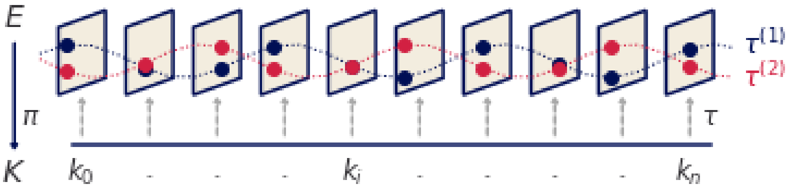


Data are sections τ on E

For any fiber bundle, there exists a map

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

s.t. $\pi(\tau(k)) = k$. $\Gamma(E)$ is the set of all global sections.



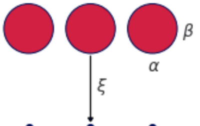
Graphic Bundle (H, S, π, D)

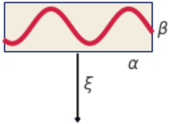
Continuity is preserved via the many s to one k map $\xi : S \rightarrow K$

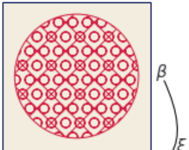
$$
\begin{tikzcd}
F \arrow[r, hook] & E \arrow[d, "\pi"'] & D \arrow[r, hook] & H \arrow[d, "\pi"'] \\
 & K \arrow[u, bend right, "\tau"'] & & S \arrow[u, bend right, "\rho"']
\end{tikzcd}
$$

$$
\begin{tikzcd}
F \arrow[r, hook] & E \arrow[d, "\pi"'] & D \arrow[r, hook] & H \arrow[d, "\pi"'] \\
 & K \arrow[u, bend right, "\tau"'] & & S \arrow[u, bend right, "\rho"'] \arrow[ll, "\xi"']
\end{tikzcd}
$$

D is a proxy for the target display, for example $(x, y, r, g, b) \in D$







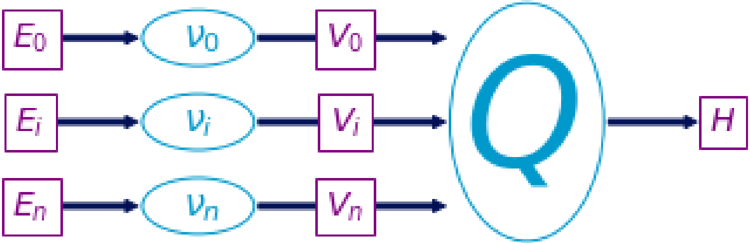
Visual bundle (V, K, π, P)

$$\mathcal{A} : \mathcal{E} \to \mathcal{H}$$

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

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

Visualization Assembly Function



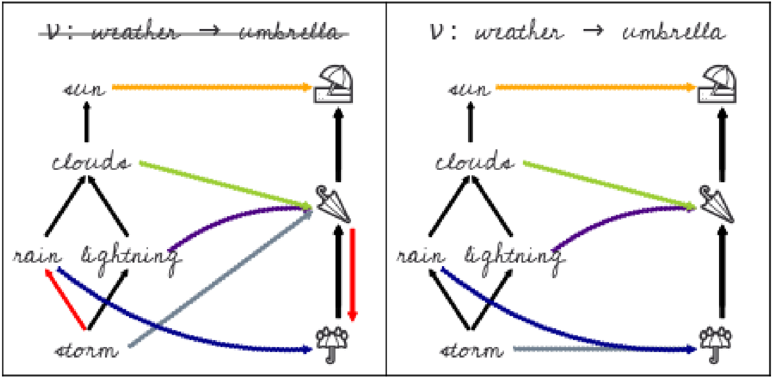
$$\{\nu_0, \ldots, \nu_n\} : \{\tau_0, \ldots, \tau_n\} \mapsto \{\mu_0, \ldots, \mu_n\}$$

$$Q = \nu \circ \tau$$

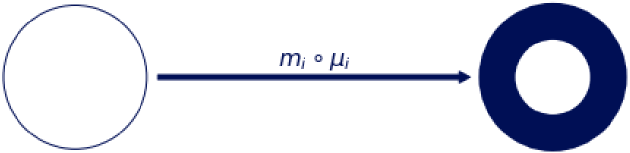
Group Equivariance: Stevens' Scales [Ste46]

scale	**group**	**constraint**
nominal	permutation	if $r_1 \neq r_2$ then $\nu(r_1) \neq \nu(r_2)$
ordinal	monotonic	if $r_1 \leqslant r_2$ then $\nu(r_1) \leqslant \nu(r_2)$
interval	translation	$\nu(x + c) = \nu(x) + c$
ratio	scaling	$\nu(xc) = \nu(x) * c$

Monoid Equivariance: Partial Orders



Visualization Equivariance

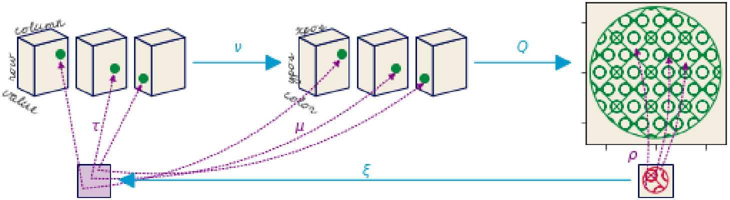


$$Q(\mu) = Q(\mu') \implies Q(m \circ \mu) = Q(m \circ \mu')$$

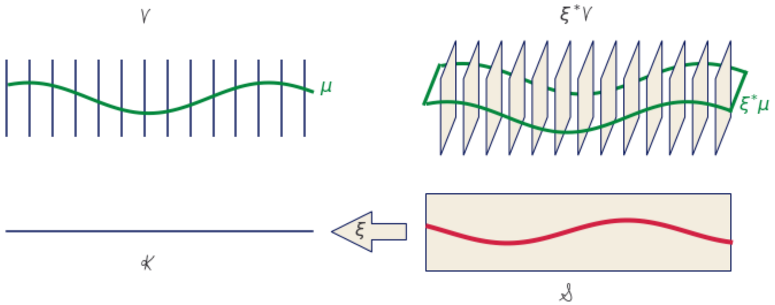
Scatter: $Q(xpos, ypos)(\alpha, \beta)$

Line: $Q(xpos, \hat{n}_1, ypos, \hat{n}_2)(\alpha, \beta)$

Image $Q(xpos, ypos, color)$

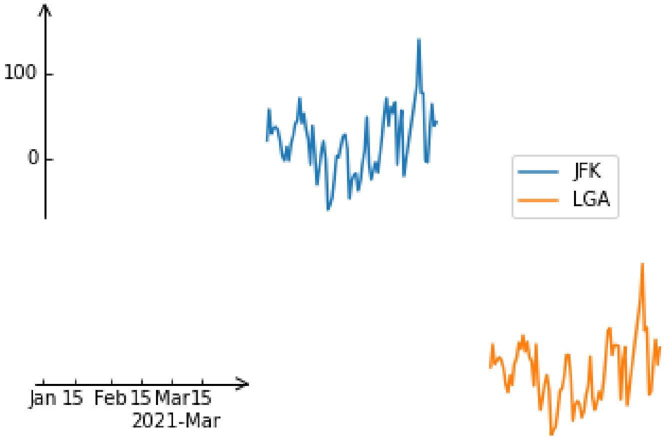


Build Q over K: $\hat{Q}$



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

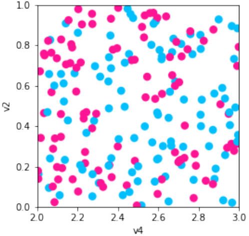
Composition of artists $+ \coloneqq \sqcup E_i$

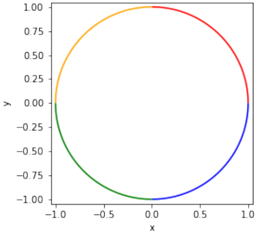


TEAM driven rearchitecture of Matplotlib

- complex visualizations
- structure preserving maps from data to visual
  - data and graphics have equivalent continuity
  - properties are equivariant under monoid actions
- fiber bundles are an abstraction
  - topologically complex heterogenous data
  - target display spaces

How do we make things?






```
fig, ax = plt.subplots()
artist = Point(data, transforms)
ax.add_artist(artist)
```

```
fig, ax = plt.subplots()
artist = Line(data, transforms)
ax.add_artist(artist)
```

ν

```python
cmap =  color.Categorical({'true':'deeppink', 'false':'deepskyblue'})
transforms = {'x': {'name': 'v4', 'encoder': lambda x: x},
                'y': {'name': 'v2', 'encoder': lambda x: x},
                'facecolors': {'name':'v3', 'encoder': cmap},
                's':{'name': None ,
                    'encoder': lambda _: itertools.repeat(.02)}}
```

- `lambda x: x` is identity ν
- `{'name':None}` map into P without corresponding τ
- `color.Categorical` is custom ν

A

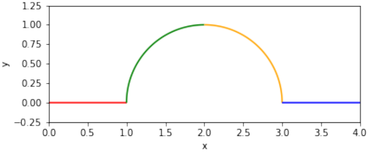
```
class ArtistClass(matplotlib.artist.Artist):
    def __init__(self, E, V, *args, **kwargs):
        # set properties that are specific to the artist
        # stash the input E and V
        super().__init__(*args, **kwargs)

    def qhat(self, **args):
        # set the properties of the graphic

    def draw(self, renderer):
        # returns tau, indexed on fiber then key
        tau = self.E.view(self.axes)
        # visual channel encoding applied fiberwise
        visual = {p_i: nu_i(tau_i)
                  for p_i, nu_i, tau)i
                  in zip(self.V, tau)}
        self.qhat(**visual)
        # pass configurations off to the renderer
        super().draw(renderer)
```

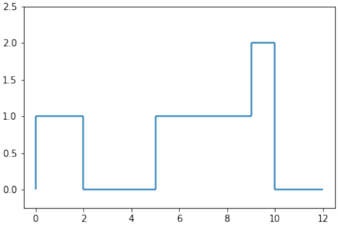
```python
class Point(mcollections.Collection):
    def assemble(self, x, y, s, facecolors='C0' ):
        # construct geometries of the circle glyphs in visual coordinates
        # set attributes of glyphs

class Line(mcollections.LineCollection):
    def assemble(self, x, y, color='C0'):
        # assemble line marks as set of segments
```

Continuity

```python
class PointData:
    # Fiberbundle is consistent across all sections
    FB = FiberBundle({'tables': ['vertex']},
        {'v1': float, 'v2': str, 'v3': float})
    def tau(self, k):
        return # tau evaluated at one point k

class LineData:
    FB = FiberBundle({'tables': ['edge']},
                 {'x' : float, 'y':  float, 'color':mtypes.Color()})
    def tau(self, k):
        return # tau evaluated on interval k
```

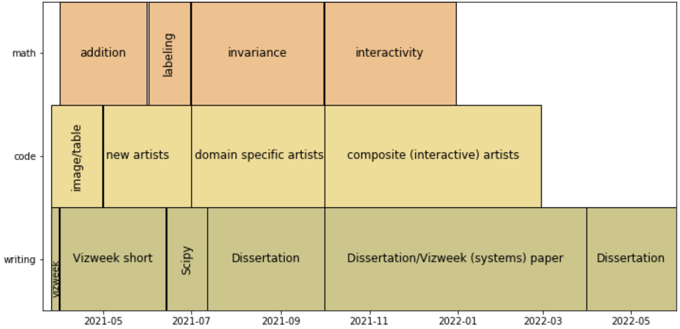
Same Artist, Different E






```
LineData(FB, edge_table, vertex_table, connect=True)
LineData(FB, edge_table, vertex_table, num_samples=2, connect=False)
```

Proposed Work



Acknowledgments

- Professor Michael Grossberg and Dr. Thomas Caswell
- Professor Haralick, Professor Vo, Professor Manovich, Dr. Hanwell
- Matplotlib development team
- CZI EOSS (grant number 2019-207333) from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation

References I

[Wic16] H. Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[SRE12] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri. "NIH Image to ImageJ: 25 Years of Image Analysis". In: *Nature Methods* 9.7 (July 2012), pp. 671–675.

[BHJ09] M. Bastian, S. Heymann, and M. Jacomy. "Gephi: An Open Source Software for Exploring and Manipulating Networks". en. In: *Proceedings of the International AAAI Conference on Web and Social Media* 3.1 (Mar. 2009).

References II

[SWH14] A. Satyanarayan, K. Wongsuphasawat, and J. Heer. "Declarative Interaction Design for Data Visualization". en. In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. Honolulu Hawaii USA: ACM, Oct. 2014, pp. 669–678.

[Stu21] S. Studies. *Culturevis/Imageplot*. Jan. 2021.

[Ell+02] J. Ellson et al. "Graphviz— Open Source Graph Drawing Tools". In: *Graph Drawing*. Ed. by P. Mutzel, M. Jünger, and S. Leipert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 483–484.

[Van+18] J. VanderPlas et al. "Altair: Interactive Statistical Visualizations for Python". en. In: *Journal of Open Source Software* 3.32 (Dec. 2018), p. 1057.

References III

[Sof+21] N. Sofroniew et al. *Napari/Napari: 0.4.5rc1*. Zenodo. Feb. 2021.

[HSS08] A. A. Hagberg, D. A. Schult, and P. J. Swart. "Exploring Network Structure, Dynamics, and Function Using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by G. Varoquaux, T. Vaught, and J. Millman. Pasadena, CA USA, 2008, pp. 11–15.

[STH02] C. Stolte, D. Tang, and P. Hanrahan. "Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases". In: *IEEE Transactions on Visualization and Computer Graphics* 8.1 (Jan. 2002), pp. 52–65.

References IV

[Han06] P. Hanrahan. "VizQL: A Language for Query, Analysis and Visualization". In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. SIGMOD '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 721.

[MHS07] J. Mackinlay, P. Hanrahan, and C. Stolte. "Show Me: Automatic Presentation for Visual Analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007), pp. 1137–1144.

[HA06] J. Heer and M. Agrawala. "Software Design Patterns for Information Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 853–860.

References V

[Dat] *Data Representation in Mayavi — Mayavi 4.7.2 Documentation*. https://docs.enthought.com/mayavi/mayavi/data.html.

[Hun07] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science Engineering* 9.3 (May 2007), pp. 90–95.

[Wt20] M. Waskom and t. seaborn development team. *Mwaskom/Seaborn*. Zenodo. Sept. 2020.

[HH17] S. Hoyer and J. Hamman. "Xarray: ND Labeled Arrays and Datasets in Python". In: *Journal of Open Research Software* 5.1 (2017).

[BOH11] M. Bostock, V. Ogievetsky, and J. Heer. "D^3 Data-Driven Documents". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (Dec. 2011), pp. 2301–2309.

References VI

[Han+15] M. D. Hanwell et al. "The Visualization Toolkit (VTK): Rewriting the Rendering Code for Modern Graphics Cards". en. In: *SoftwareX* 1-2 (Sept. 2015), pp. 9–12.

[Gev+12] B. Geveci et al. "VTK". In: *The Architecture of Open Source Applications* 1 (2012), pp. 387–402.

[RV11] P. Ramachandran and G. Varoquaux. "Mayavi: 3D Visualization of Scientific Data". In: *Computing in Science Engineering* 13.2 (Mar. 2011), pp. 40–51.

[BJ09] Brian Wylie and Jeffrey Baumes. "A Unified Toolkit for Information and Scientific Visualization". In: *Proc.SPIE*. Vol. 7243. Jan. 2009.

[AGL05] J. Ahrens, B. Geveci, and C. Law. "Paraview: An End-User Tool for Large Data Visualization". In: *The visualization handbook* 717.8 (2005).

References VII

[TM04] M. Tory and T. Moller. "Rethinking Visualization: A High-Level Taxonomy". In: *IEEE Symposium on Information Visualization*. 2004, pp. 151–158.

[Mac86] J. Mackinlay. "Automating the Design of Graphical Presentations of Relational Information". In: *ACM Transactions on Graphics* 5.2 (Apr. 1986), pp. 110–141.

[Tuf01] E. R. Tufte. *The Visual Display of Quantitative Information*. English. Cheshire, Conn.: Graphics Press, 2001.

[Nor93] D. A. Norman. *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

References VIII

[BB92] D. M. Butler and S. Bryson. "Vector-Bundle Classes Form Powerful Tool for Scientific Visualization". en. In: *Computers in Physics* 6.6 (1992), p. 576.

[BP89] 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.

[Spi10] D. I. Spivak. *Databases Are Categories*. en. Slides. June 2010.

[Spi] D. I. Spivak. "SIMPLICIAL DATABASES". en. In: (), p. 35.

[Mun14] T. Munzner. *Visualization Analysis and Design*. AK Peters Visualization Series. CRC press, Oct. 2014.

[Ste46] S. S. Stevens. "On the Theory of Scales of Measurement". In: *Science* 103.2684 (1946), pp. 677–680.

[KS14] G. Kindlmann and C. Scheidegger. "An Algebraic Process for Visualization Design". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (Dec. 2014), pp. 2181–2190.

[Mac87] J. Mackinlay. "Automatic Design of Graphical Presentations". English. PhD Thesis. Stanford, 1987.

[Wil05] L. Wilkinson. *The Grammar of Graphics*. en. 2nd ed. Statistics and Computing. New York: Springer-Verlag New York, Inc., 2005.

[SSS09] T. Sugibuchi, N. Spyratos, and E. Siminenko. "A Framework to Analyze Information Visualization Based on the Functional Data Model". In: *2009 13th International Conference Information Visualisation*. 2009, pp. 18–24.

References X

[VFR13] P. Vickers, J. Faith, and N. Rossiter. "Understanding Visualization: A Formal Approach Using Category Theory and Semiotics". In: *IEEE Transactions on Visualization and Computer Graphics* 19.6 (June 2013), pp. 1048–1061.

[nLa21] nLab authors. "Action". In: (Mar. 2021).

[Ghr14] R. W. Ghrist. *Elementary Applied Topology*. Vol. 1. Createspace Seattle, 2014.

[Ghr18] R. Ghrist. "Homological Algebra and Data". In: *Math. Data* 25 (2018), p. 273.

[Ceg19] A. M. Cegarra. "Cohomology of Monoids with Operators". In: *Semigroup Forum*. Vol. 99. Springer, 2019, pp. 67–105.

[ZK09] C. Ziemkiewicz and R. Kosara. "Embedding Information Visualization within Visual Representation". In: *Advances in Information and Intelligent Systems*. Ed. by Z. W. Ras and W. Ribarsky. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 307–326.

Mathematical Models of Vizualization

algebraic process data and viz transforms are symmetric [KS14]

$$\begin{tikzcd} E \arrow[r, "\nu"] \arrow[d, "m"'] & V \arrow[r, "Q"] & H \arrow[d, "\varphi(m)"] \\ E \arrow[r, "\nu"'] & V \arrow[r, "Q"'] & H \end{tikzcd}$$

language APT and GoG: syntax, semantics, and grammar [Mac86; Mac87; Wil05]

functional dependencies relationship between components [SSS09]

category theory *understanding* = *read* ∘ *render* [VFR13]

Fiber is all possible values a variable can be [Spi10; Spi]

Given a space of all possible values $\mathbb{U}$

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

a fiber component is the restricted space $\mathbb{U}_{\sigma(c)}$.

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

DT data types of the variables in the dataset

$\mathbb{U}$ disjoint union of all values of type $T \in \mathbf{DT}$

C variable names, $c \in C$

$\mathbb{U}_\sigma$ $\mathbb{U}$ restricted to the data type of a named variable

Monoid actions

A monoid M is a set with

associative binary operator $*: M \times M \to M$

identity element $e \in M$ such that $e * a = a * e = a$ for all $a \in M$.

left monoid action

A set F with an action[nLa21] $\bullet : M \times F \to F$ with the properties:

associativity for all $f, g \in M$ and $x \in F$, $f \bullet (g \bullet x) = (f * g) \bullet x$

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

Keeping track of sections with sheafs

Restriction maps of a sheaf describe how local $\iota^*\tau$ can be glued into larger sections [Ghr14; Ghr18]

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

The inclusion map $\iota : U \to K$ pulls E over U such that the pulled back $\iota^*\tau$ only contains records over $U \subset K$.

Rendering: Define a Pixel

Given a pixel

$$p = \left[y_{top}, y_{bottom}, x_{right}, x_{left}\right]$$

the inverse map of the bounding box

$$S_p = {\rho_{xy}}^{-1}(p)$$

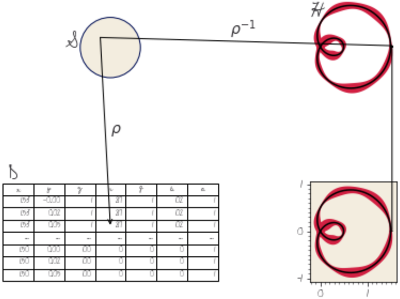
is a region $S_p \subset S$ such that

$$r_p = \iint_{S_p} \rho_r(s) ds^2 \qquad (1)$$

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

$$b_p = \iint_{S_p} \rho_b(s) ds^2 \qquad (3)$$

yields the color of the pixel.



$\mathcal{A} : \mathcal{E} \to \mathcal{E}$

The topological artist is a sheaf map

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

that carries homomorphism of monoid actions $\varphi : M \to M'$ [Ceg19]

$$A(m \cdot r) = \varphi(m) \cdot A(r)$$

Visual Channel Encoders

We define the visual transformers ν on components of the data bundle τ_i

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

as the set of equivariant maps with the constraint

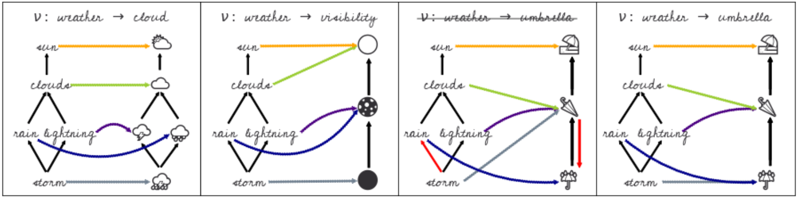
$$\nu_i(m_r(E_i)) = \varphi(m_r)(\nu_i(E_i))$$

where $\varphi : M \to M'$ carries a homomorphism of monoid actions.

P Components

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

Monoid Equivariance: Partial Orders



Glyph

The glyph is the graphic generated by $Q(S_j)$ where the path connected components $J \subset K$ are defined

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

such that the path γ from k to j is a continuous function from the interval [0,1] and S_j is the region

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

such that the glyph is differentiable, in keeping with Ziemkiewicz and Kosara's description of a glyph[ZK09].

Artist Equivalance class

When artists share a base space

$$K_2 \hookrightarrow K_1$$

a composition operator can be defined such that the the artists can be considered to be acting on different components of the same section.

Complex ν

```python
class Categorical:
    def __init__(self, mapping):
        # check that the conversion is to valid colors
        assert(mcolors.is_color_like(color) for color in mapping.values())
        self._mapping = mapping

    def __call__(self, value):
        # convert value to a color
        return [mcolors.to_rgba(self._mapping[v]) for v in values]
```

That we can test for action equivariance

```python
def test_nominal(values, encoder):
    m1 = list(zip(values, encoder(values)))
    random.shuffle(values)
    m2 = list(zip(values, encoder(values)))
    assert sorted(m1) == sorted(m2)
```

Artist

```python
class ArtistClass(matplotlib.artist.Artist):
    def __init__(self, data, transforms, *args, **kwargs):
        # properties that are specific to the graphic
        self.data = data
        self.transforms = transforms
        super().__init__(*args, **kwargs)

    def assemble(self, **args):
        # set the properties of the graphic

    def draw(self, renderer):
        # returns K, indexed on fiber then key
        view = self.data.view(self.axes)
        # visual channel encoding applied fiberwise
        visual = {p: t['encoder'](view[t['name']])
                    for p, t in self.transforms.items()}
        self.assemble(**visual)
        # pass configurations off to the renderer
        super().draw(renderer)
```

Artists: Scatter & Line

```
class Point(mcollections.Collection):
    def assemble(self, x, y, s, facecolors='C0' ):
        # construct geometries of the circle glyphs in visual coordinates
        self._paths = [mpath.Path.circle(center=(xi,yi), radius=si)
                        for (xi, yi, si) in zip(x, y, s)]
        # set attributes of glyphs, these are vectorized
        # circles and facecolors are lists of the same size
        self.set_facecolors(facecolors)

class Line(mcollections.LineCollection):
    def assemble(self, x, y, color='C0'):
        #assemble line marks as set of segments
        segments = [np.vstack((vx, vy)).T for vx, vy in zip(x, y)]
        self.set_segments(segments)
        self.set_color(color)
```

View

```python
def view(self, axes):
    table = defaultdict(list)
    for k in self.keys:
        table['index'].append(k)
        for (name, value) in zip(self.FB.fiber.keys(), self.tau(k)[1]):
            table[name].append(value)
    return table
```

`VertexSimplex` (name, value), value is scaler

`EdgeSimplex` (name, value), value is [x0, ..., xn]

Fiber Bundle

```
@dataclass
class FiberBundle:
"""
Attributes
----------
K: {'tables': []}
F: {variable name: type}
"""
    K: dict
    F: dict
```

GraphLine Data Model

```
class GraphLine:
    def __init__(self, FB, edge_table, vertex_table, num_samples=1000,
                        connect=False):
        #set args as attributes and generate distance
        if connect: # test connectivity if edges are continuous
            assert edge_table.keys() == self.FB.F.keys()
            assert is_continuous(vertex_table)

    def tau(self, k):
        # evaluates functions defined in edge table
        return(k, (self.edges[c][k](self.distances)
                        for c in self.FB.F.keys()))

    def view(self, axes):
        # walk the edge_vertex table to return the edge function
        table = defaultdict(list)
        for (i, (start, end)) in sorted(zip(self.ids, self.vertices),
                                        key=lambda v:v[1][0]):
            table['index'].append(i)
            # same as view for line, returns nested list
            for (name, value) in zip(self.FB.F.keys(), self.tau(i)[1]):
                table[name].append(value)
        return table
```