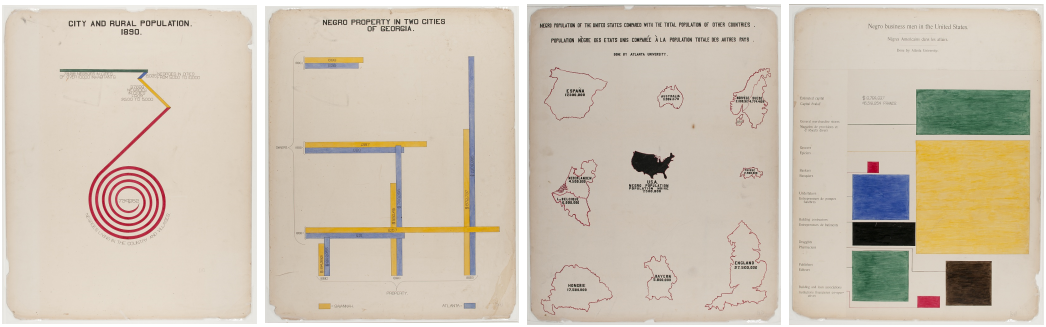
1 Introduction

In data visualization it is assumed the graphical representation of data match the properties of the data, and in this work we propose that the mathematical notion of equivariance formalizes this expectation. We meld the infoviz communities interest in heterogenous often discrete datasets with the scientific visualization communities emphasis on continuous and sometimes topologically complex datasets. To demonstrate the practical value of our model, we propose a model driven re-architecture of the artist layer of the Python visualization library Matplotlib. In addition to providing a way to ensure the library preserves structure, we propose a functional approach to improve modularity, maintainability, and point to ways in which the library could better support concurrency and interactivity.

2 Background



(a) (b) (c) (d)

Figure 1: Du Bois' data portraits[13] of post reconstruction Black American life exemplify that the fundemental characteristics of data visualization is that the visual elements vary in proportion to the source data. In figure 1a, the length of each segment maps to population; in figure 1b, the bar charts are intersected to show the number of property owners and how much they own in a given year; in figure 1c the countries are scaled to population size; and figure 1d is a treemap where the area of the rectangle is representative of the number of businesses in each field. The images here are from the Prints and Photographs collection of the Library of Congress [1, 2, 30, 31]

This work aims to develop a model of visualization such that a tool built using this model could support visualizations as varied as those of Du Bois in figure 1; to do so, we first discuss the criteria by which a visualization is evaluated. Byrne et al. propose that visualizations have graphic representations that are mappings from data to visuals and figurative representations that have meaning due to their similarity in shape to external concepts [8]. In figure 1c, Du Bois combines a graphical representation where glyph size varies by population with a figurative representation of those glyphs as the countries the data is from, which means that the semantic and numerical properties of the data are preserved in the graph. Tufte specifies that visual representations must be in proportion to the quantitative data being represented for a chart to be faithful and that there should be no extra information in the graphic or figurative elements of the graph, but otherwise his notion of graphic integrity

is heavily context dependent[33]. As is Norman's Naturalness Principal, which states that visualizations are more understandable when the properties of the representation match the properties of the information being represented[22]. Bertin takes it as a given that data properties match visual properties, so much so that Munzner argues it is inherently built into his classification system [21] which is displayed in figure 2.

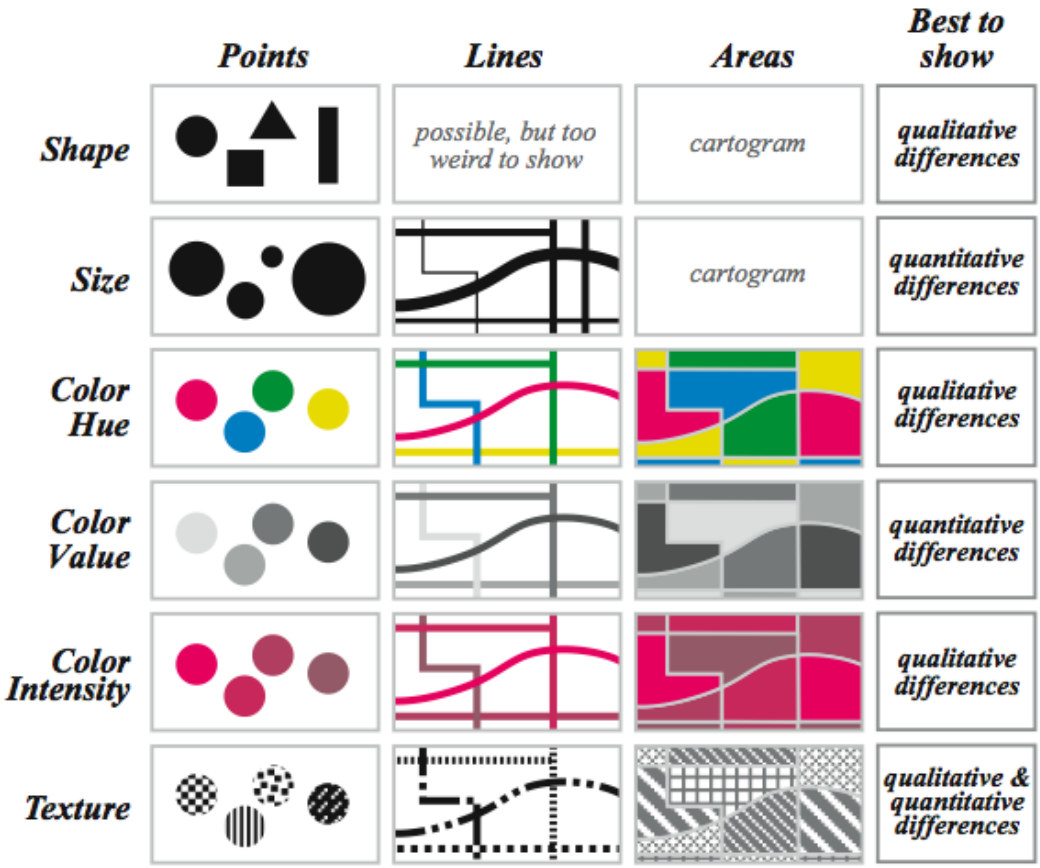


Figure 2: Retinal variables are a codification of how position, size, shape, color and texture are used to illustrate variations in the components of a visualization. The best to show column describes which types of information can be expressed in the corresponding visual encoding. This tabular form of Bertin's retinal variables is from Understanding Graphics [19] who reproduced it from *Making Maps: A Visual Guide to Map Design for GIS* [16]

As described by Mackinlay, a visualization tool produces a graphical design and an image rendered based on that design. He defines the graphical design as the set encoding relations from data to visual representation[17], and the design rendered in an idealized abstract space is what throughout this paper we will refer to as a graphic. Mackinlay proposes that a visualization tool's expressiveness is a measure of how much of the structure of the data the tool encodes, while the tools effectiveness describes how much design choices are made in deference to perceptual saliency [9–11, 21]. Mackinlay's definition of expressiveness is

formalized at the visual encoding level, which as shown in figure 2 refers to the components of a graphic such as the color or position of a glyph. Bertin first classified these graphic components as retinal variables and discussed which types of data they can express [3] and how they are composited on point, line, and area graphical marks, as shown in figure 2 correspond. Marks can be generalized to glyphs, which are graphical objects that convey one or more attributes of the data entity mapped to it[35]. Mackinlay's expressiveness criteria is well defined for the visual variables, such that he suggests the viability of a strict encoding relation that is a homomorphic mapping which preserves some binary operator from one domain to another [18]. We expand on this suggestion by proposing that monoid action equivariance is a strict condition of building valid encoders. Mackinlay does not provide a generalized criteria for plot types, instead embedding the requirements within the definition of the charts.

2.1 Data

Tory and Möller propose that assumptions about the structure of data are built into the visual algorithms that display that information [32]. Specifically they note that discrete and continuous data and their attributes form a discipline independent design space [23].

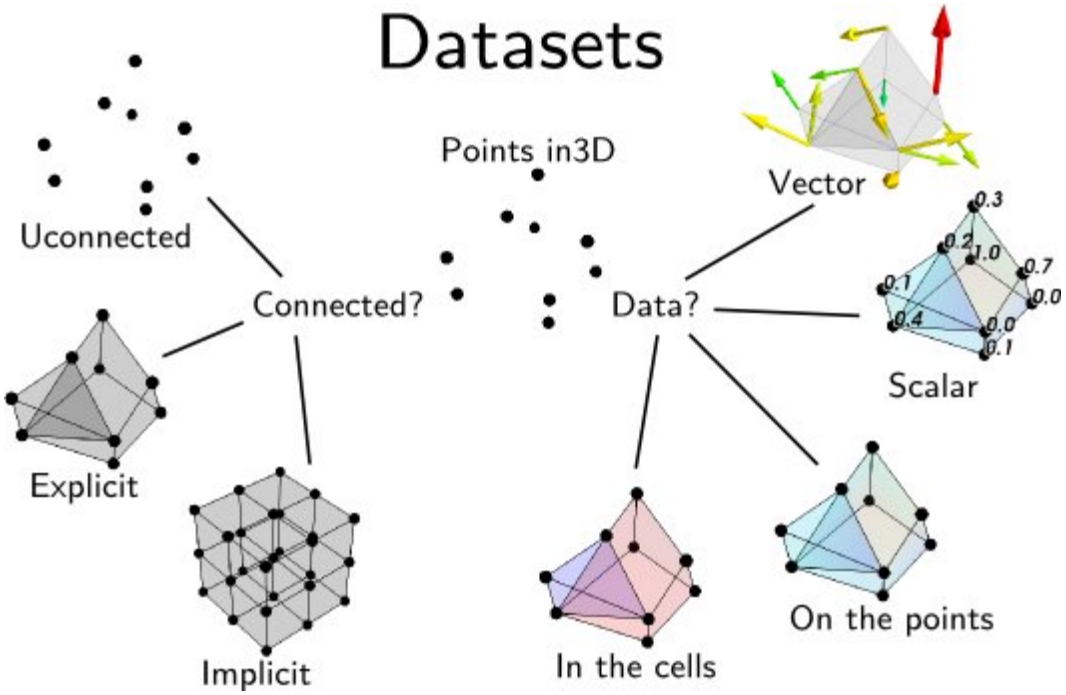


Figure 3: One way to describe data is by the connectivity of the points in the dataset. A database for example is often discrete unconnected points, while an image is an implicitely connected 2D grid. This image is from the Data Representation chapter of the MayaVi 4.7.2 documentation.[12]

As shown in figure 3, there are many types of continuity in data. A database typically consists of unconnected records, while an image is an implicit 2D grid and a network is some

sort of explicitly connected graph. In this work we will refer to the points of the dataset as *records* to indicate that a point can be a vector. Each *component* of the record is a single object, such as a temperature, a color, or an image. The way in which these records are connected is the *connectivity*, *continuity*, or more generally *topology*.

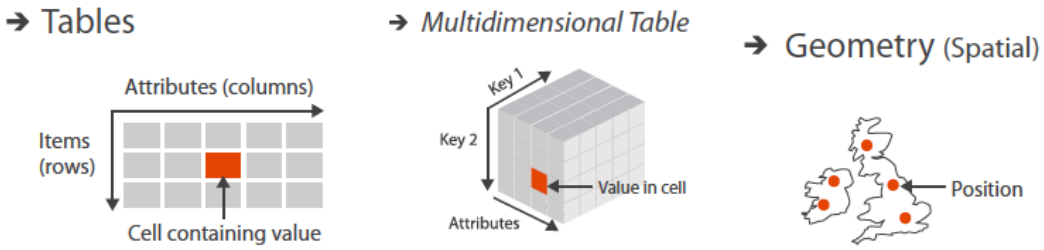


Figure 4: Keys are unique lookup values used to find individual observations in the dataset. Keys are positional references, and can be coordinates on a map or unique values such as a primary key in a database or a (time, latitude, longitude) index in a data cube. Image modified from a diagram from Munzner's *Visualization Analysis and Design* [21]

Often this topology has metadata associated with it, as shown in figure 4. Munzner denotes this metadata as *keys* that can be used to locate the record, and further to locate the specific *value* [20]. In figure 4 tables have keys identifying the row and column and sometimes also the depth, while maps have a location key. We propose that information rich keys such as location are additional components of the record, and that instead there are coordinate free structural keys that identify the location of the record within a dataset of any continuity.

In this work, we extend Butler's topology driven representation of visualization data [6, 7]. Butler proposes that fiber bundles are a good model for visualization data because it allows for encoding the connectivity seperately from the records and supports discrete and ND continuous datasets. Since Butler's model lacks a robust way of describing the components of the record, we fold in Spivak's Simplacial formulation of databases [27, 28] so that we can encode a schema like discription of the data in the fiber bundle. We then propose criteria on expressivity that take into account both the components and the continuity of the data.

2.2 Tools

A motivator for this work is that currently Matplotlib carries implicit assumptions about data continuity in how each function interfaces with the input data. This work proposes a unified internal representation that encodes connectivity in a common interface. Matplotlib aims to natively support data of varying connectivities, so tools primarily concerned with visualizing relational data of the type found in databases are insufficient models. Many of these tools are built on top of Wilkenson's Grammar of Graphics [37] which itself incorporates Mackinlay's A Presentation Tool (APT) display algebra; GoG derivative include ggplot[36], protovis[4] and D3 [5], vega[25] and altair[34]. While many of these tools support images, the first class data container is a table like object of discrete records. Tools that primarily support images are also insufficient. For example ImageJ[26] and the ImagePlot[29]macro have some support for visualizing non image components of a complex

data set, but mostly in service to the image being visualized. Plugins exist, but must work around the everything is an image data model[38]. There are also visualization tools that do not carry implicit assumptions about structure, for example vtk[14, 15] and its derivatives such as MayaVi[24]. Not totally positive but I think VTK is deeply coupled to the renderer and that's why we're not using it as a model, but not positive and this is literally what Marc worked on...

2.3 contribution

The contribition of this work is a model we call the topological artist model (TAM) in which data and graphics can be viewed as sections of fiber bundles. This model allows for (1) decomposing the translation of data fields (variables) into visual channels via an equivariant map on the fibers and (2) a topology-preserving map of the base spaces that translates the dataset connectivity into graphical elements. Furthermore, this model supports an algebraic sum operation such that more complex visualizations can be built from simple ones. To demonstrate the practical value of the model, we built a prototype where we represent the topological base spaces using triangulation, make use of programming types for the fiber, and build on Matplotlib's existing infrastructure for the rendering.