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: This collection of visualizations is the inside cover art of W. E. B. Du Bois's Data Portraits[2]

1 Introduction

Why am I starting with DuBois? Visualization library - spells out/regularizes the ways you can aesthetically encode data, preserving structure of data, scatter - magnitude between values preserved - some ways an accurate representation of the data, concerned w/ as data is changed, graph has to change accordingly, graph needs to change accordingly - owns responsibility for preserving measurement integrity when specified by encodings - specify what you want to preserve - (choose which) preserve relations in a strict math/set theory sense - functional is a good fit b/c it spells out these constraints Drawing program - user is responsible for preserving these measurements

Some of DuBois visualizations are common chart types most modern visualization tools (cite excel, tableu, matplotlib, ggplot, maybe high chart) support, while others are far more

custom and likely need tools with drawing capabilities (cite: matplotlib, base r, d3) [2]. This intentionality in what charts the tool supports and the flexibility it gives the user underpins why we care about this problem. Drawing programs don't really have to care about the data structure because everything is explicit - the user chooses the shape, line, color, they are by hand manually doing all the encoding. Visualization libraries on the other hand allow users to specify which bits of the data they want encoded and how, but there's a lot of implicit mapping of the raw variables in the data to the encodings. We want some confidence that the visualization tool is making the mapping we intended. We often do this by implictely encoding the data structure in the grammar of the visualiztion tool (grammaer of graphics, vega, ggplot, altair), but a goal of Matplotlib [**huntermatplotlib2007**] is to be fairly data structure agnostic. We are proposing an architecture that converts these implicit assumptions into explicit contracts between the data and the visualization.

Why functional? because what we care about is the interface, the contract between the data and the visualization. We need the minimal amount of information that fulfills our contract that we can then put on the screen. (insert react notes here). Our theoretical framework is that that visualization is a transformation from one topological space to another to a CW complex (this probably needs to be explained more), and a functional programming paradigm allows us to directly express that in code.

1.1 What will be the work

Matplotlib needs to support use cases across a vast range of science domains, enabling complex visualizations [1]. At the same time, common visualizations in a domain need to be fluid for the end-practitioners, tuned to the domain's standard data structures, and with domain-specific customization options exposed. To achieve both of these goals, we need to continue to foster a two-layer ecosystem with a shared core (Matplotlib) and many domain-specific libraries (seaborn, nipy, ...).

In the current grant cycle we have been developing a new architecture that is heavily invested in cleanly separating the three steps in a visualization pipeline: data representation, encoding the data as visual elements, and rendering those elements to screen. We believe

that a better delineation of these steps will allow domain practitioners to more easily implement extensions. Following on the work done this year to design consistent data and artist abstractions, we will develop simpler and more expressive user-level APIs in core Matplotlib and in collaboration with domain-specific libraries.

Why topology?

The difference between a line plot and a scatter plot is the former assumes that the data is continous, the latter that it is discrete (cite line and scatter - friendly?) In concrete implementation terms, matplotlib's imshow assumes that the data is continous and therefore does implementation, the matshow assumes it is discrete so it does not. And there are visualization types like area charts that allow even more information to be encoded bewteen the lines. When redesigning the architecture, we needed a way to articulate the differences between the different plot types and found that a topological approach allowed us to encode connectivity (discrete versus continous), dimensionality (point, line, area), and how much information the visualization encodes.

What even is data? Remake this graph notated w/ graph semantics:

What even is encoding? Bertin table on 69 rewrite here clearly

When talking about encodings, we are referring to Bertin's codification of the properties of the graphic system [1] seen in fig **??** as visual variables. In this system, there are the marks, which are the point, line, and area geometric primatives on the visual plane that each represent a single instance of the data being visualized [**munznerMarksChannels2014**, 1]. These marks can be varied along different visual channels size, value, texture, color, orientation, shape, and position [**munznerMarksChannels2014**, 1]. As shown in fig **??**, the marks encode topological connectivity such that they have the following graph semantics:

Points - 0d, no edges

Lines - 1d, every vertex has ¡=2 edges (1d simplex)

Area - 2d, every vertex has ¡=4 edges (2d simplex)

The visual channels in fig **??** are recommended based on the measurement type, quantative versus qualitativem and Munzner generalizes the quantative channels as magnitude

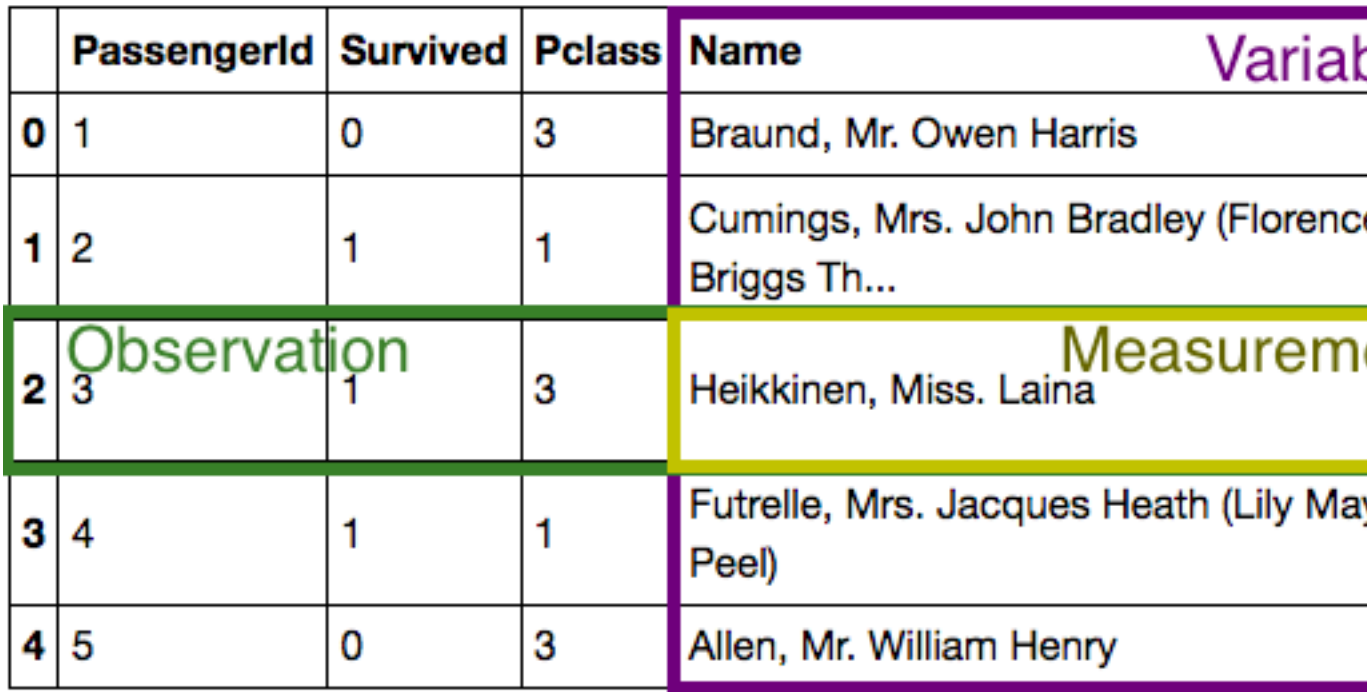

Figure 2: Modified from Munzner's Visualizarion Analysis and Design

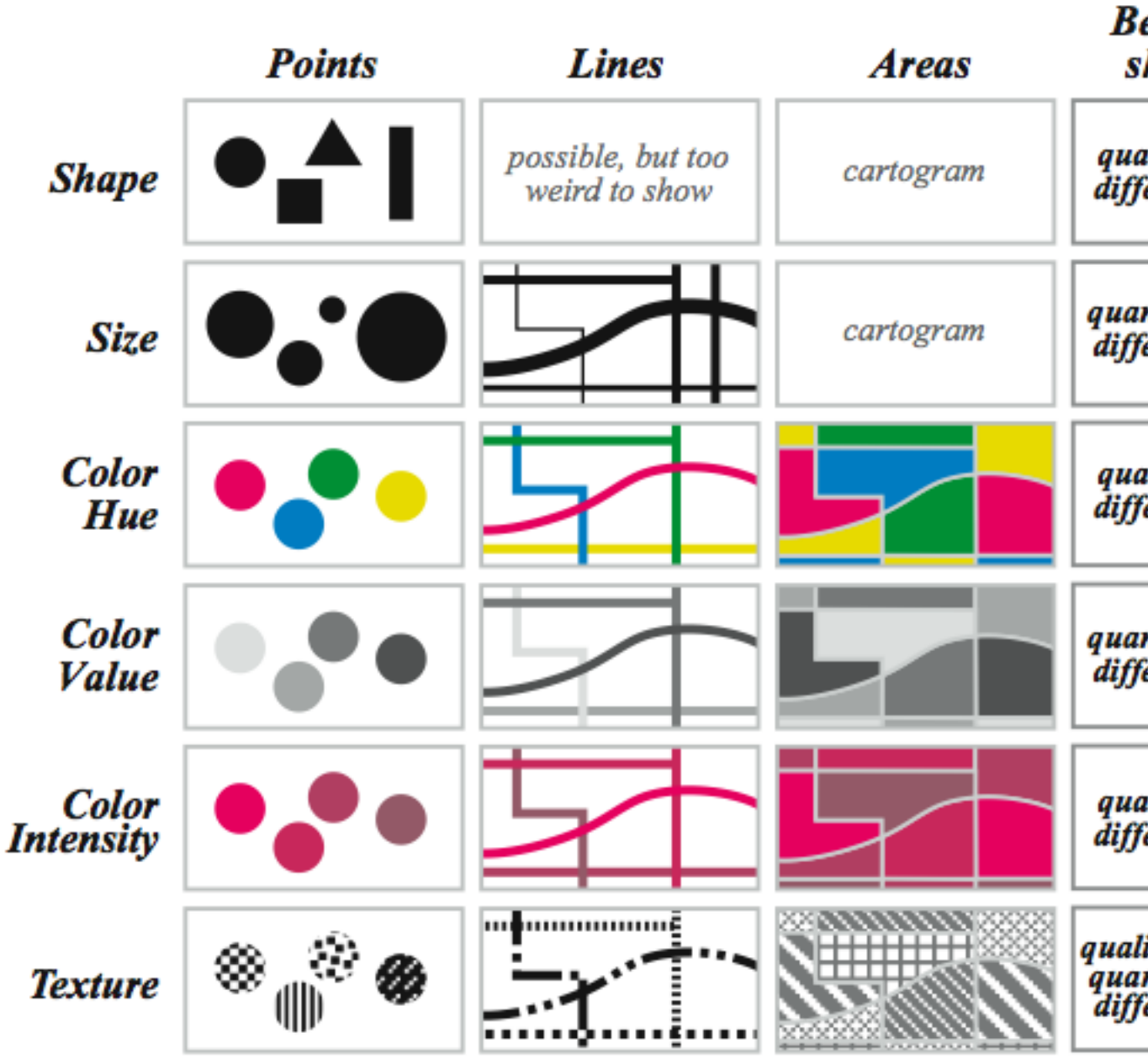

Figure 3: This organization of Bertin's retinal variables [**bertinSemiologyGraphicsDiagrams2011**] that lays out the geometry(marks) and corresponding aesthetic variations (channels) with a recommendation based on data type (qualitative versus quantative) comes from Krygier and Wood's Making Maps [3]

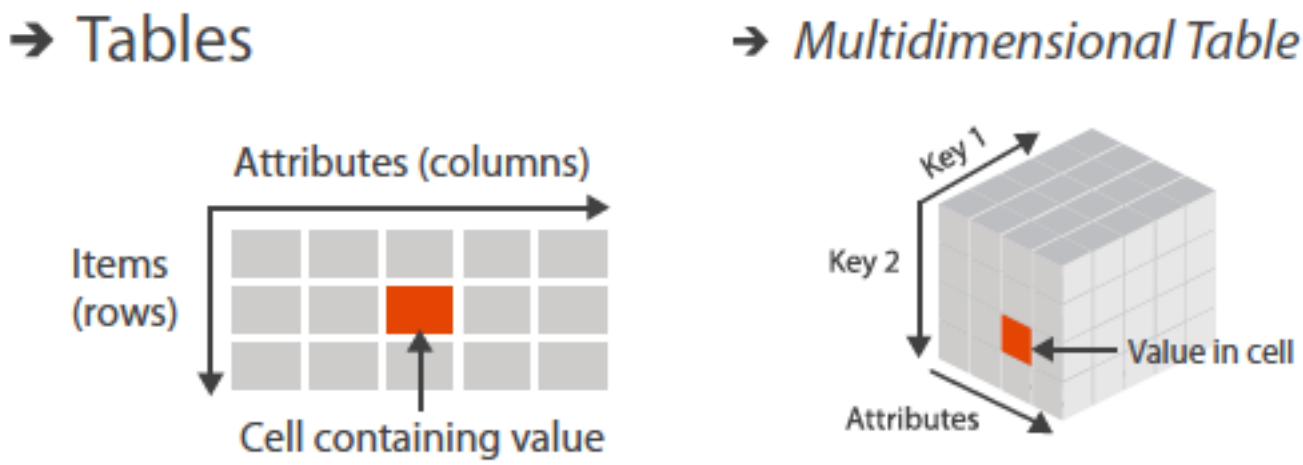

Figure 4: Keys are unique lookup values used to find individual observations in the dataset. Keys are positional references, and can be coordinates on a map or unique values such as a primary key in a database or a (time, latitude, longitude) index in a data cube. Image modified from a diagram from Munzner's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

channels for ordered attributes (measurements drawn from an ordered set of possible values) and the qualitative channels as identity channels for categorical attributes (random set of possible values). topology in measurement types, and that topology informs the channels appropriate for those types: quantative continouse, quantative discrete, qualitative catorical ordinal, qualitative categorical marks - topology of measurements (people wearing dresses) channels - topology of space from which measurements are drawn (possible dress sizes) categorical - discrete points w/o anything to do categorical - poset topology?

base space - topology of observations (people wearing dresses) fiber topology - topology of measurements (possible dress sizes)

functional 'cause focus on transforms - if transforms preserve linearity than whole pipeline does. Dev gets guarantee on calling composite.

Add table of which sorts of invariance a channel will preserve....

Munzner's key/value semantics[5] provide a way to identify variables that in effect act as metadata for other variables, such as how when we are interested in the temperature at a time the temperature is the lookup value. But that did not generalize well for complex

heterogenous datasets. In this proposal, we refine those ideas by using topology to formally describe the connectivity between the measurements.

```
make any stupid plot you want in robust & rigourous way
        * need rich description langauge + right choice of paradigm (functional)
        * ability to mathemetaically formalize/conceptual framework for doing this
        * data -> artist, need to preserved the structure of the data moreso than the data
            * how points are connected to each other
    * targeted implementation rather than protocal (numpy)
        * visualization libraries bound to the datastructures (concrete implementation)
            * encode a lot of assumptions about data in the data structure
            * MPL assume x/y plotted in order you want them in
            * topology makes the assumptions explicit
            * explicit->math->functional
    * spell out the layers of visualization libaries:
        * Data + computation -> visualization composites -> drawing library
        * domain specific library
        * utility library <- formalize this piece (SciPy Diagram)
    * viz - Munzner + Bertin
    * functional programming makes sense
```

2 Library Review

2.1 Grammar of Graphics & ggplot

Specifies the end chart (graphic) we want to specify the transformations .... [**wilkinsonGrammarGraphicsStatistics**
charts - words - instances of much more general objects (geometric primatives) - histogram =/= bar chart graphics - statements expliceltly OO 1. specification data - operations/computations trans - variable transforms (rank) scale - scale transforms coord - coordinate system element - marks and channels guide - meta elements - (axes, legends,

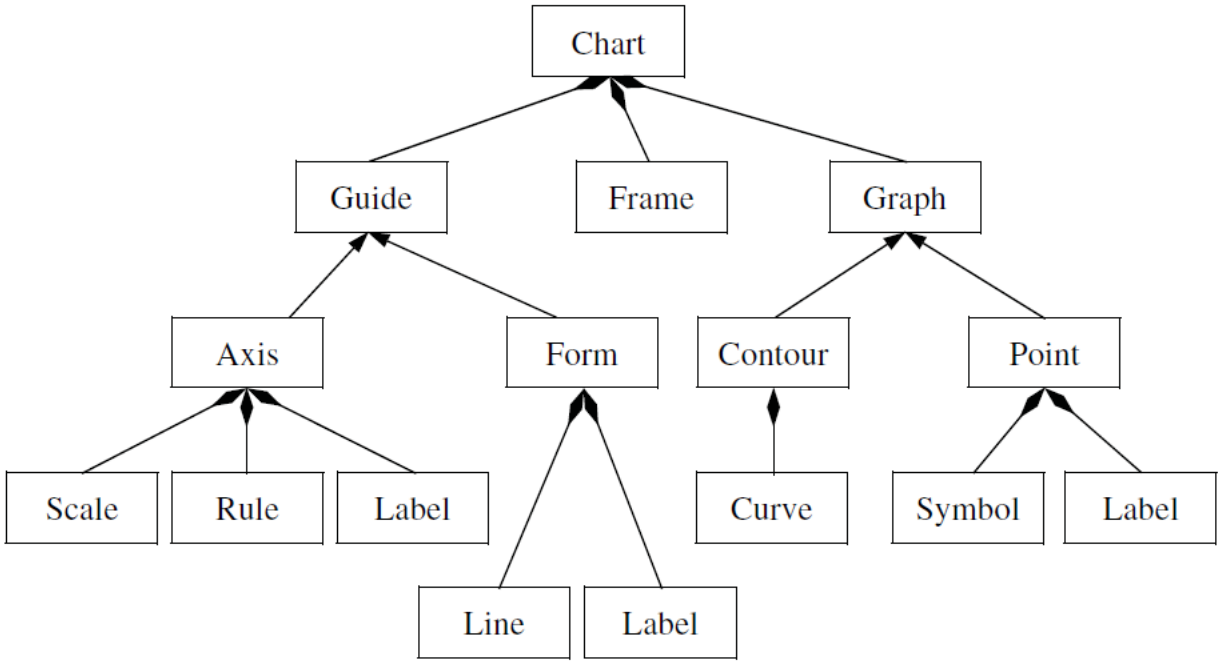

Figure 5: page 10 (introduction)

DATA: longitude, latitude = *map*(*source*("World"))
TRANS: bd = *max*(birth-death, 0)
COORD: *project.mercator*()
ELEMENT: *point*(*position*(lon*lat), *size*(bd), *color*(*color.red*))
ELEMENT: *polygon*(*position*(longitude*latitude))

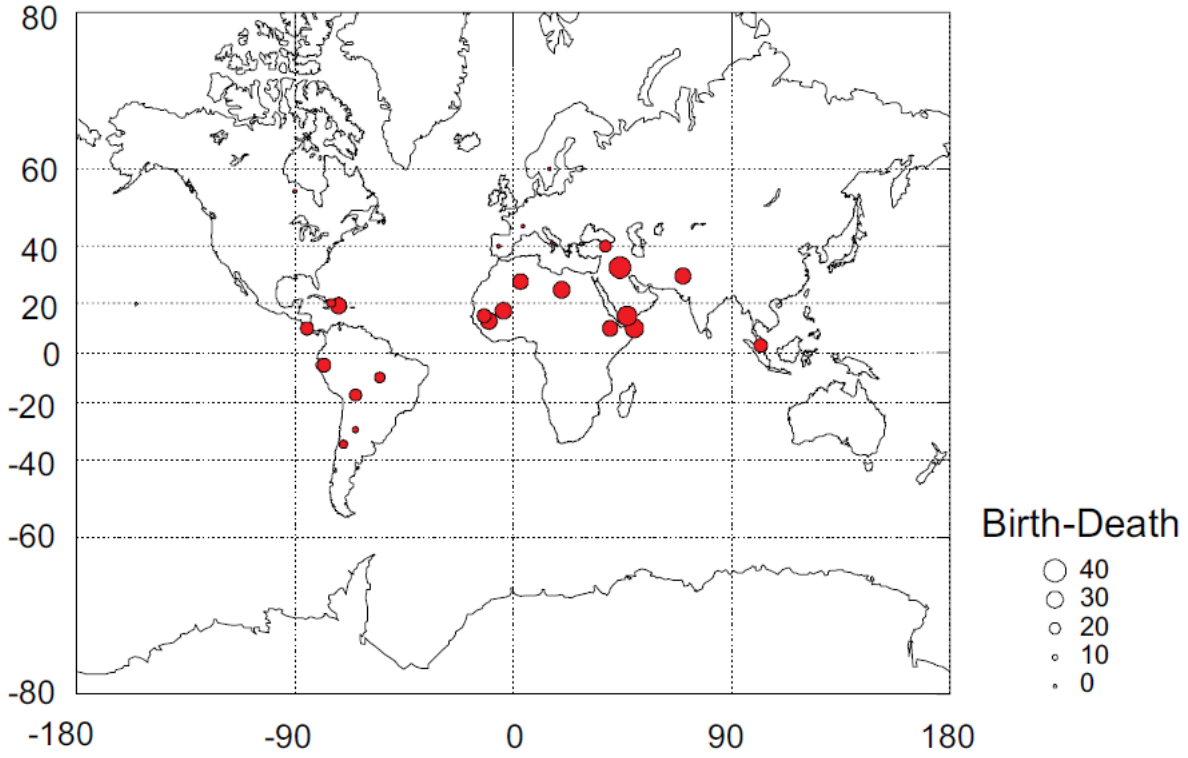

Figure 1.5 *Excess birth (vs. death) rates in selected countries*

Figure 6: Wilkenson decomposed a graphic into ...

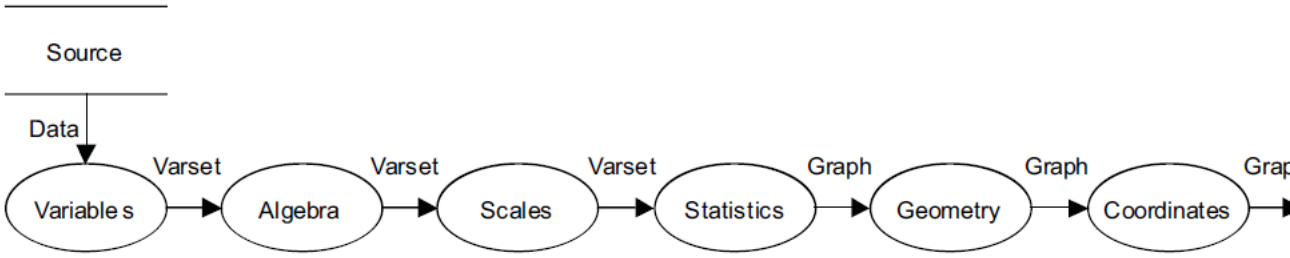

Figure 7: Diagram of Grammar of Graphics stages of visualization. Fig 2.2 from chapter 2 of The Grammar of Graphics[6]

etc) 2. assembly - kinda what happens in artist (spec-¿something that can go to renderer) 3. display - rendering

How is our proposal different? seperation in spec between what's data, aesthetics, and render specific, and what part of the architecture owns those operations

ggplot - strictly hierarchal components (layers), matplotlib - transforms that mostly happen indepently "designing and producing statistical graphics is not an art" - ties in w/ difference between drawing program and visualization library, preservation of properties of measurement type and observation type

[**wickhamGgplot2ElegantGraphics2016**]

history of viz - collins, friendly

Wilkenson suggests that the graphics pipeline could be implemented functionally but does not push it in any direction.

ordering: processed before plotted, scales before stats, marks before channels In grammar of graphics, as shown in figure 7 data is extracted from the database one variable at a time with an associated index (primary key). The algebra stage joins the variables to become the varset, which is a flat table where the columns are the variables/attributes, each row is an observation/item, and each cell contains a measurement [4]. Wilkenson then describes

the constraints of the measurement space for each variable through scales [6] with implicit assumptions of the data constraints. After this step, data is transformed computationally in service of the visualization. Some of the variables are then mapped into geometric marks (symbols). Position based visual mappings such as x, y, or height are applied at this stage. These mappings are then transformed into the coordinate system of the target graph.

Wilkenson's thoughts on coherancy are equivalent to invariance but less? formally stated ...

"To call these charts meainingul, defenders must falsify specific assumptions of GoG" [7]

gog implementationss:

1. SPSS nViZn
2. tableu
3. ggplot
4. protoviz3̧
5. vega (interactive GoG)

References

[1] 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.

[2] T. W. E. B. Du Bois Center at the University of Massachusetts, W. Battle-Baptiste, and B. Rusert. *W. E. B. Du Bois's Data Portraits: Visualizing Black America*. Princeton Architectural Press, 2018. ISBN: 978-1-61689-706-2.

[3] John Krygier and Denis Wood. *Making Maps: A Visual Guide to Map Design for GIS*. English. 1 edition. New York: The Guilford Press, Aug. 2005. ISBN: 978-1-59385-200-9.

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

[5] 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.

[6] Leland Wilkinson. *The Grammar of Graphics*. en. 2nd ed. Statistics and Computing. New York: Springer-Verlag New York, Inc., 2005. ISBN: 978-0-387-24544-7.

[7] Leland Wilkinson. *The Mathematical Foundation of Analytic Visualizations*. Georgia Tech, Apr. 2010.