Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco

Dominik Moritz @domoritz

with Chenglong Wang, Greg Nelson, Halden Lin, Adam Smith, Bill Howe, Jeff Heer
Designing Visualizations can be Tedious

Barley Data

<table>
<thead>
<tr>
<th>Variety</th>
<th>Site</th>
<th>Age</th>
<th>Yield</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manchuria</td>
<td>University Farm</td>
<td>4</td>
<td>27</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Waseca</td>
<td>4</td>
<td>48</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Morris</td>
<td>5</td>
<td>27</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Crookston</td>
<td>3</td>
<td>39</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Grand Rapids</td>
<td>6</td>
<td>32</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Duluth</td>
<td>5</td>
<td>28</td>
<td>1931</td>
</tr>
</tbody>
</table>
### Designing Visualizations can be Tedious

Selecting Data Fields

*Barley Data Fields*

<table>
<thead>
<tr>
<th>Variety</th>
<th>Site</th>
<th>Age</th>
<th>Yield</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manchuria</td>
<td>University Farm</td>
<td>4</td>
<td>27</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Waseca</td>
<td>4</td>
<td>48</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Morris</td>
<td>5</td>
<td>27</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Crookston</td>
<td>3</td>
<td>39</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Grand Rapids</td>
<td>6</td>
<td>32</td>
<td>1931</td>
</tr>
<tr>
<td>Manchuria</td>
<td>Duluth</td>
<td>5</td>
<td>28</td>
<td>1931</td>
</tr>
<tr>
<td>Glabron</td>
<td>University Farm</td>
<td>4</td>
<td>48</td>
<td>1931</td>
</tr>
</tbody>
</table>
Designing Visualizations can be Tedious

Barley

Select Data Fields

Site Yield

Transform Data

SUM(Yield) BY Site

Data Fields

Data

Transformed Data

<table>
<thead>
<tr>
<th>Site</th>
<th>SUM(Yield)</th>
</tr>
</thead>
<tbody>
<tr>
<td>University Farm</td>
<td>653</td>
</tr>
<tr>
<td>Waseca</td>
<td>962</td>
</tr>
<tr>
<td>Morris</td>
<td>708</td>
</tr>
<tr>
<td>Crookston</td>
<td>748</td>
</tr>
<tr>
<td>Grand Rapids</td>
<td>498</td>
</tr>
<tr>
<td>Duluth</td>
<td>559</td>
</tr>
</tbody>
</table>
Designing Visualizations can be Tedious...

Barley Data

Select Data Fields

Site Yield

Transform Data

SUM(Yield) BY Site

Transformed Data

Design Encoding

Visual Encoding

Transformed Data

Graph showing yield by site:
- Crookston
- Duluth
- Grand Rapids
- Morris
- University Farm
- Waseca

Yield values range from 0 to 1,000.
...and requires design expertise.
Automated Visualization Design

Systems that use design guidelines to support users create good visualizations.

Different metrics and only inferable goals.

Humans and machines work together.
Automated Visualization Design

Systems that use design guidelines to support users create good visualizations.

Presentation Tool

- Incomplete Specification
- Graphical Design
- Visualization

synthesize render

APT Mackinlay
SAGE Mittal et al.
BOZ Casner
ShowMe Mackinlay et al.
VizDeck Key et al.
SeeDB Vartak et al.
Voyager (2) Wongsuphasawat et al.
What is Draco?

A formal model that represents visualizations as sets of logical facts and design guidelines as a collection of hard and soft constraints over these facts.
Outline

Modeling Visualization Design

Applying Visualization Design

Learning Visualization Design
Modeling Visualization Design in Draco

Visualizations $\rightarrow$ Logical Facts

Design Knowledge $\rightarrow$ Constraints
Visualizations → Logical Facts

- **data** = barley
- **mark** = bar

**encoding1** =
- **channel** = x
- **field** = yield
- **type** = continuous
- **aggregate** = sum
- **zero** = ⊤

**encoding2** =
- **channel** = y
- **field** = site
- **type** = categorical
1. Domain of Attributes

"The *mark* of a chart should be one of *bar*, *line*, *area* or *point*."
Design Knowledge → Constraints

1. Domain of Attributes

mark $\in \{\text{bar, line, area, point}\}$
channel $\in \{x, y, \text{color}, \text{text, opacity, shape}\}$
field $\in \{\text{site, variety, yield, year, age}\}$
type $\in \{\text{categorical, continuous}\}$
aggregate $\in \{\text{sum, count, mean, median}\}$
zero $\in \{\bot, \top\}$
bin $\in \mathbb{N}$

...
2. Integrity Constraints

“Only *continuous* fields can be *aggregated*”

\[ \forall e \in \text{Encodings} : e \cdot \text{aggregate} \Rightarrow e \cdot \text{type} = \text{continuous} \]
2. Integrity Constraints

\[ \forall e \in \text{Encodings} : \ e \cdot \text{aggregate} \Rightarrow e \cdot \text{type} = \text{continuous} \\
(\exists e \in \text{Encodings} : e \cdot \text{channel} = \text{shape}) \Rightarrow \text{mark} = \text{point} \\
\text{mark} = \text{bar} \Rightarrow (\exists e \in \text{Encodings} : (e \cdot \text{channel} = x \lor e \cdot \text{channel} = y)) \\
\ldots \]
3. Preferences

“Prefer to include zero.”

Cost if soft constraint is violated

\[ 5 \forall e \in \text{Encodings} : e \cdot \text{zero} = T \]
3. Preferences

“Prefer to include **zero** for **continuous** fields.”

Cost if soft constraint is violated

\[ 5 \forall e \in \text{Encodings} : e \cdot \text{type} = \text{continuous} \Rightarrow e \cdot \text{zero} = T \]
Visualizations → Logical Facts

Design Knowledge → Constraints

1. Domain of Attributes
2. Integrity Constraints
3. Preferences (Soft Constraints)
Source of Rules: CompassQL

Recommendation Engine for Voyager & Voyager 2

Years of tuning and experience

[Wongsuphasawat et al.]
Outline

Modeling Visualization Design

Applying Visualization Design

Learning Visualization Design
Automated Visualization Design

Verifying Designs

Visualization Recommendation

Autocomplete

Comparing Visualization Models

Enumerating Design Space

Constrain Models

Synthesis

Recommendation
Automated Visualization Design

Verifying Designs

Autocomplete

Enumerating Design Space

Visualization Recommendation

Comparing Visualization Models

Constrain Models

Synthesis
In Draco, we formulate Automated Visualization Design as finding the model that optimally completes the constraints.
"I want a visualization that shows **site** and **yield** from the **barley dataset.**"

data = barley \land \\
\exists e : e.f.\text{field} = \text{site} \land \\
\exists e : e.f.\text{field} = \text{yield} \\
1. Domain of Attributes \\
2. Integrity Constraints \\
3. Preferences \\
\begin{align*}
\text{Clingo solver}
\end{align*}

\text{encoding1} = \begin{cases} 
\text{channel} = x \\
\text{field} = \text{yield} \\
\text{type} = \text{quantitative} \\
\text{zero} = \top 
\end{cases} \\
\text{encoding2} = \begin{cases} 
\text{channel} = y \\
\text{field} = \text{site} \\
\text{type} = \text{categorical} 
\end{cases}
data = barley ∧
∃e : e.field = yield
data = barley ∧
∃e : e.field = yield
data = barley \land \\
\exists e : e \cdot \text{field} = \text{yield}

data = barley \land \\
\exists e : e \cdot \text{field} = \text{yield} \land e \cdot \text{bin}
data = barley \land \\
\exists e : e \text{. field} = \text{yield}

\[
\begin{align*}
data &= \text{barley} \land \\
\exists e : e \text{. field} &= \text{yield} \land e \text{. bin}
\end{align*}
\]

1. not $e \in \text{Encoding}$
2. $\forall e \in \text{Encoding} : e \text{. aggregate} \neq \text{count}$
3. $\exists e : e \text{. continuous} = \top \Rightarrow \exists e \in \text{Encoding} : e \text{. aggregate}$

Adding count prevents overlap
data = barley ∧
∃e : e.field = yield

data = barley ∧
∃e : e.field = yield ∧
∃e : e.field = site

data = barley ∧
∃e : e.field = yield ∧
∃e : e.field = site ∧
mark = bar
data = barley ∧
∃e : e.field = yield ∧
∃e : e.field = site ∧
mark = bar
Example: Draco CQL

CompassQL
~20 lines of code per rule
+ Scoring logic

Draco-CQL
70 Hard Constraints
110 Soft Constraints
Example: Draco CQL

**CompassQL**
~20 lines of code per rule
+ Scoring logic

**Draco-CQL**
70 Hard Constraints
110 Soft Constraints

GOOD!!!!!!
Example: Draco CQL

**CompassQL**
~20 lines of code per rule
+ Scoring logic

**Draco-CQL**
70 Hard Constraints
110 Soft Constraints

Wait... 😐
Example: Draco CQL

**CompassQL**
~20 lines of code per rule
+ Scoring logic

**Draco-CQL**
70 Hard Constraints
110 Soft Constraints

Wait… 😞

How to weight rules?
Classical AI
Classical AI + Machine Learning
Outline

Modeling Visualization Design

Applying Visualization Design

Learning Visualization Design
Learning Visualization Design in Draco

Features
Violations of Soft Constraints

Training Data
Pairs of Ranked Visualizations

Learning Algorithm
Learning to Rank with Linear SVM

\[
L = \frac{1}{n} \sum_{i=1}^{k} \max(0, 1 - y_i w^T(x_{i1} - x_{i2})) + \lambda \|w\|_2
\]

\[
w^* = \arg \min_w L
\]

\[v_1, v_2, \ldots, v_k\]

\(v_i\): the number of violations of rule i.
Violations of Soft Constraints as Features

Can express non-linear relationships even though the learning system is linear
Learning Design Knowledge

Feature Vector
positive example
\[ [u_1, u_2, \ldots, u_k] \]

Feature Vector
negative example
\[ [v_1, v_2, \ldots, v_k] \]

\[ \arg \max_w \sum_{i \in 0\ldots k} w_i (u_i - v_i) \]

\( w \) is the weight vector of the soft constraints

\( v_i \): the number of violations of constraint \( i \)
Example: Draco-Learn

Draco-CQL
(no task awareness)

Value

Task

Summary
Example: Draco-Learn

Draco-CQL
(no task awareness)
+
2 User Studies of
Task-Vis Correlation
[Kim et al. 2018, Saket et al. 2018]
=
Draco-Learn
(task-aware)

- ignore weights
- 1,100 Vis Pairs
- 96% Test Accuracy
(65% for Draco-CQL)
Example: Draco-Learn

<table>
<thead>
<tr>
<th>Task</th>
<th>Value</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draco-CQL (no task awareness)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Draco-Learn (task aware)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Future Directions
Future Directions

Automated Design for Interactive Dashboards
Future Directions

Automated Design for Interactive Dashboards

Integration in Plotting APIs

https://altair-viz.github.io/
Future Directions

Automated Design for Interactive Dashboards
Integration in Plotting APIs
Tools to Browse, Update, and Compare Draco Knowledge Bases
Evaluate impact of new perceptual models

uwdata.github.io/draco-editor
Draco Meetup

Thursday 13:00 at Shaan
meet at 12:50 at the revolving doors
Draco

Extensible and Adaptive Knowledge Base of Visualization Design

Automated Visualization Design Tool

Formal Reasoning

Shared Resource for Vis Community

Accelerate Knowledge Transfer

Learn Visualization Design

Dominik Moritz @domoritz et al.

Draco: Formalizing Visualization Design Knowledge as Constraints

Draco is a formal framework for representing design knowledge about effective visualization design as a collection of constraints.

You can use Draco to find effective visualization designs in Vega-Lite. Draco’s constraints are implemented in based on Answer Set Programming (ASP) and solved with the Clingo constraint solver. Draco can learn weights for the recommendation system directly from the results of graphical perception experiments.