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

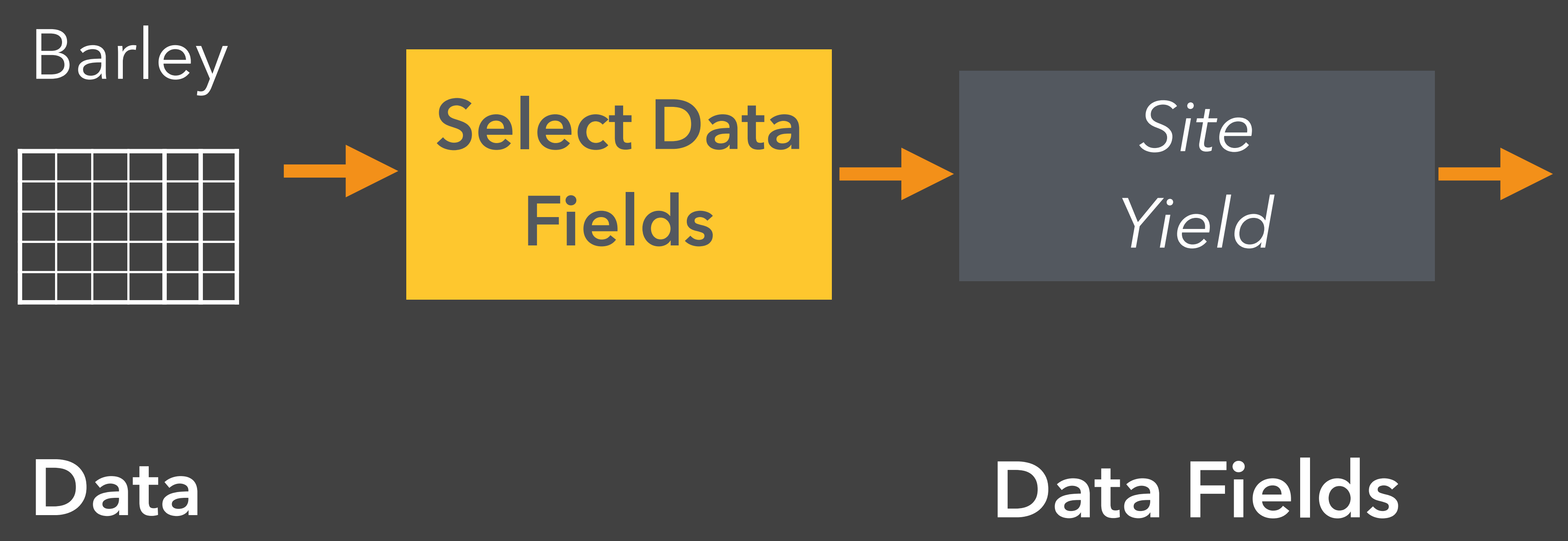
Designing Visualizations can be Tedious

Barley

Data

Variety	Site	Age	Yield	Year
Manchuria	University Farm	4	27	1931
Manchuria	Waseca	4	48	1931
Manchuria	Morris	5	27	1931
Manchuria	Crookston	3	39	1931
Manchuria	Grand Rapids	6	32	1931
Manchuria	Duluth	5	28	1931
Glabron	University Farm	4	48	1931

Designing Visualizations can be Tedious



Variety	Site	Age	Yield	Year
Manchuria	University Farm	4	27	1931
Manchuria	Waseca	4	48	1931
Manchuria	Morris	5	27	1931
Manchuria	Crookston	3	39	1931
Manchuria	Grand Rapids	6	32	1931
Manchuria	Duluth	5	28	1931
Glabron	University Farm	4	48	1931

Designing Visualizations can be Tedious

Barley → Select Data Fields → *Site Yield* → **Transform Data** → *SUM(Yield) BY Site*

Data

Data Fields

Transformed Data

Site	SUM(Yield)
University Farm	653
Waseca	962
Morris	708
Crookston	748
Grand Rapids	498
Duluth	559

Designing Visualizations can be Tedious…

Barley

Data

Select Data Fields

Site
Yield

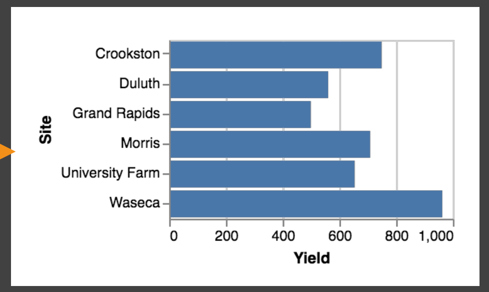
Data Fields

Transform Data

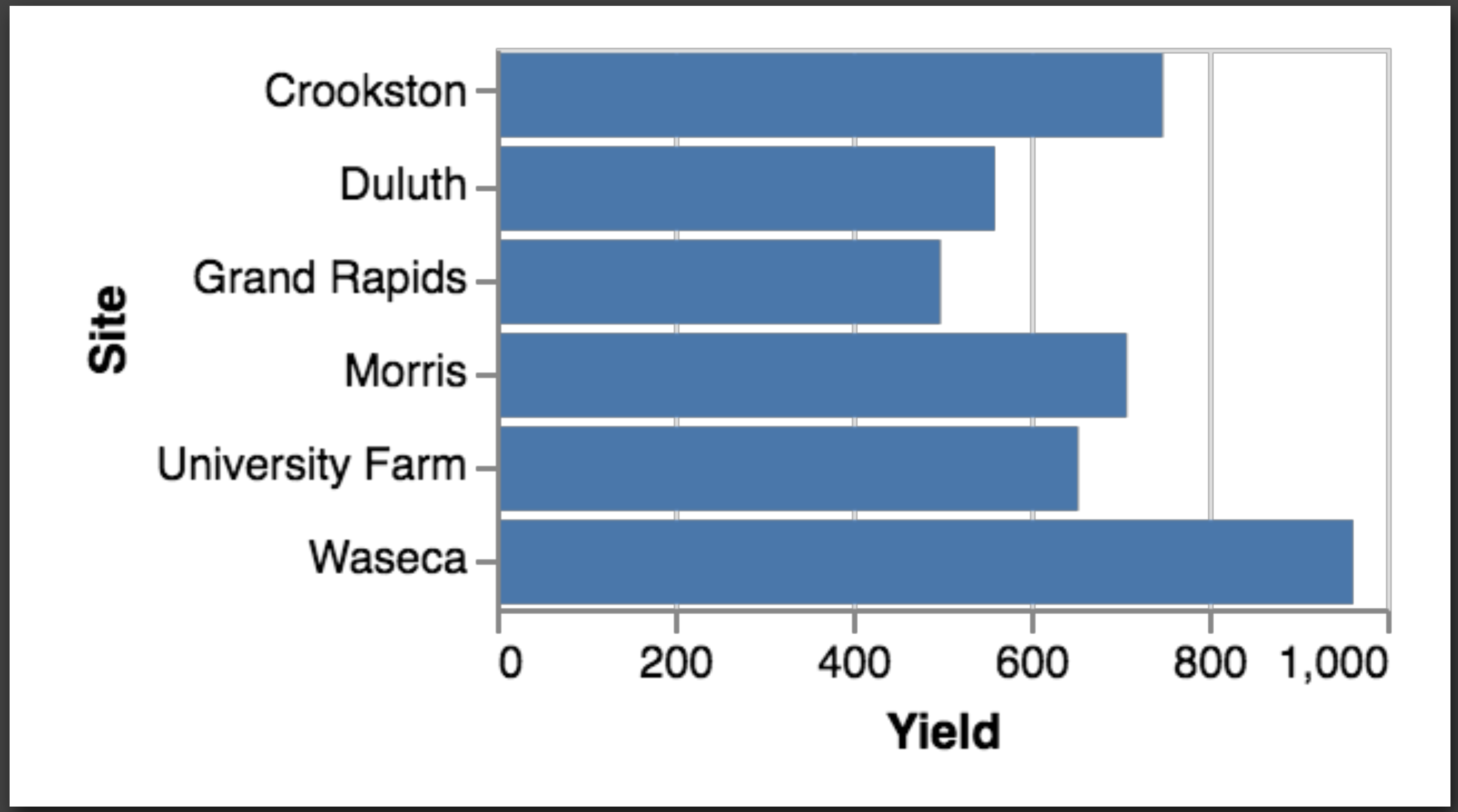
SUM(Yield)
BY *Site*

Transformed Data

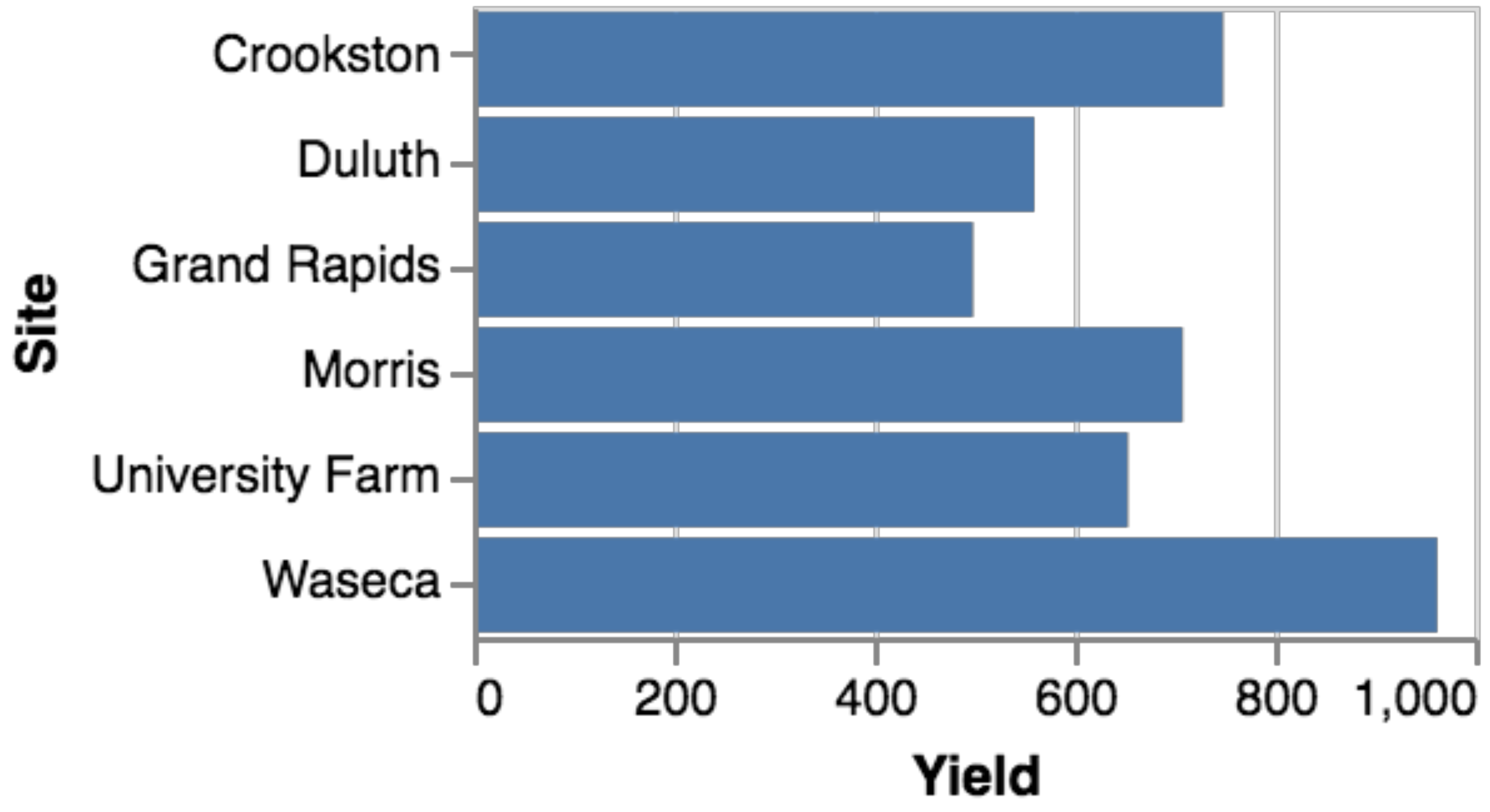
Design Encoding



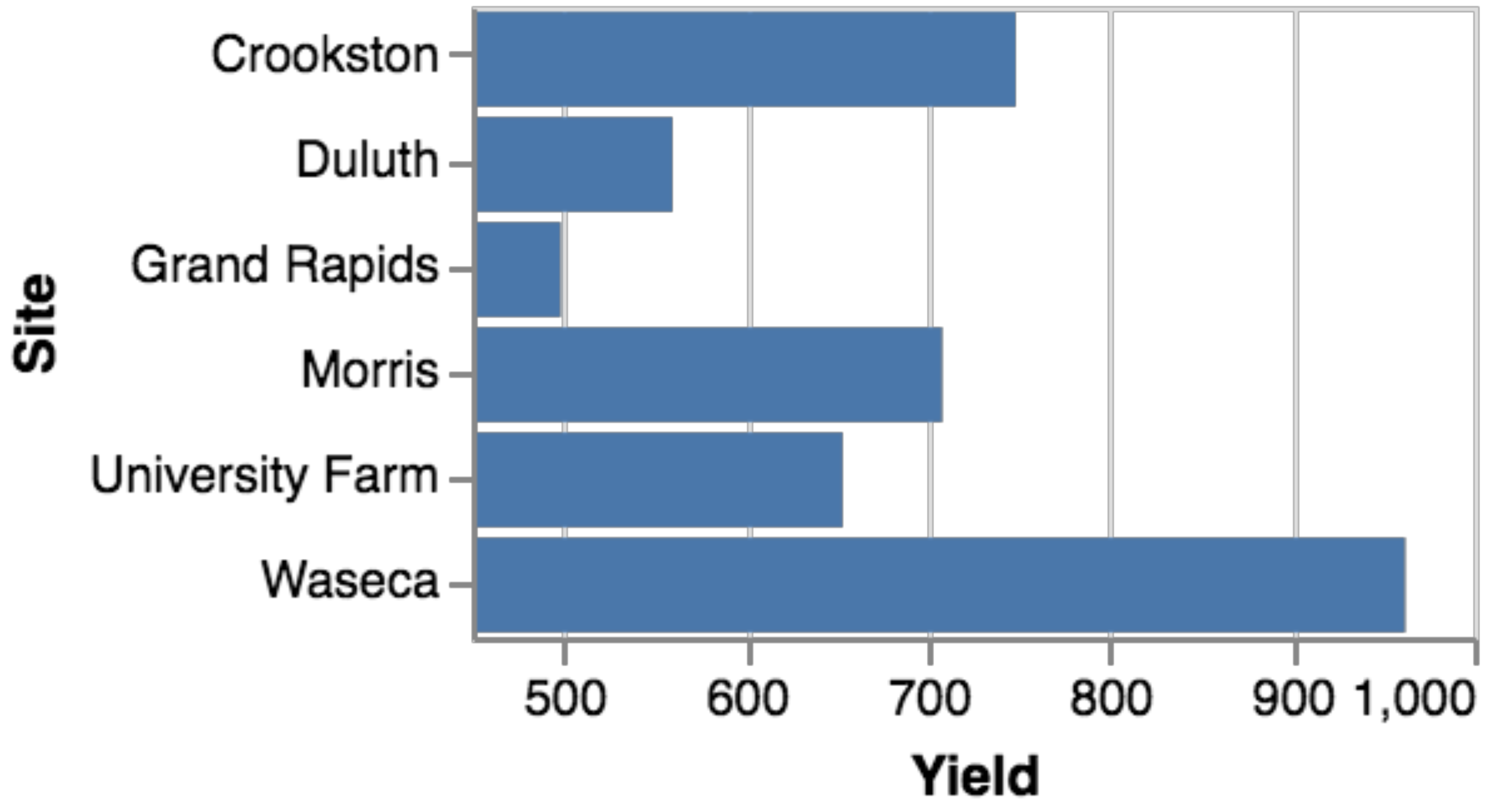
Visual Encoding

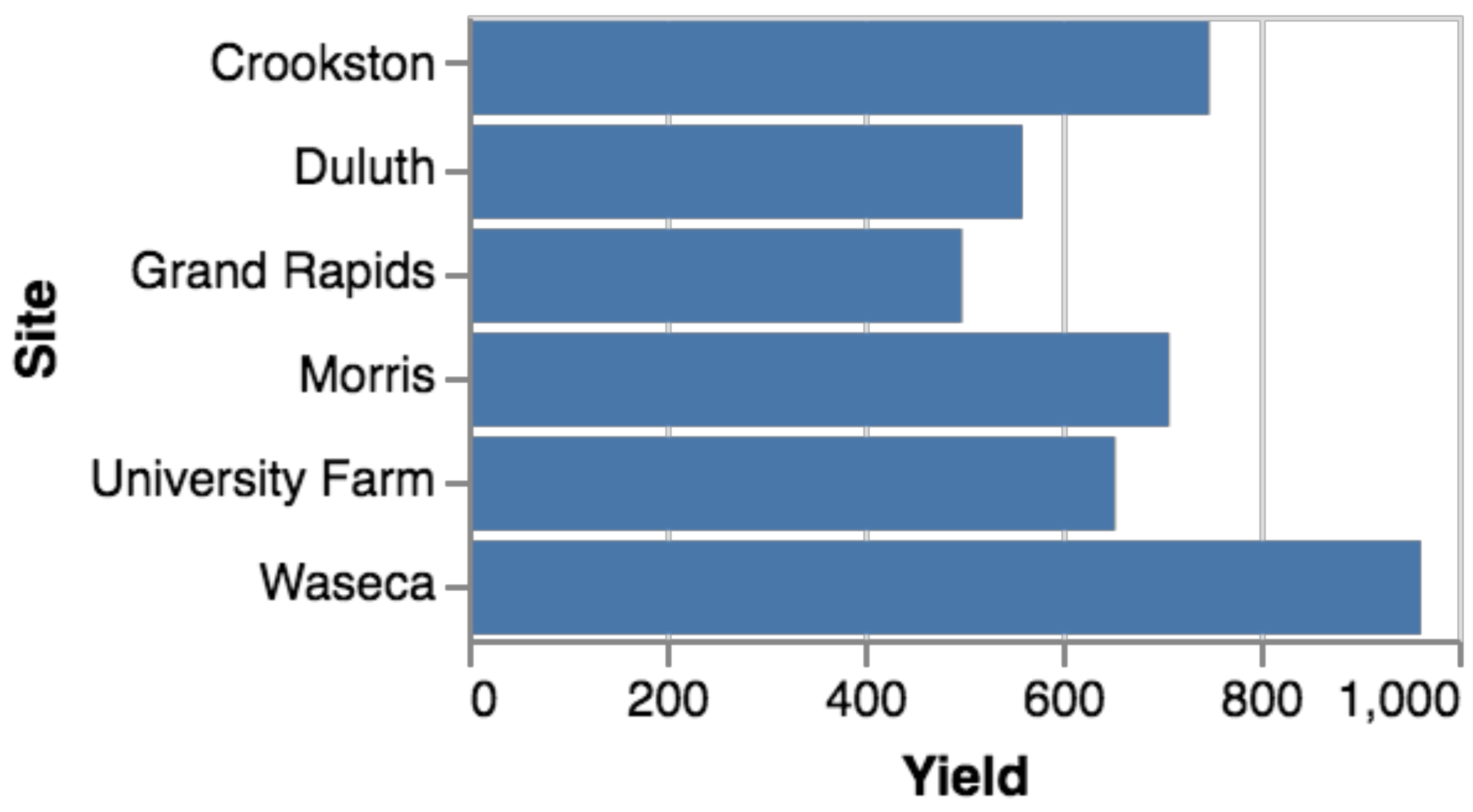


…and requires design expertise.

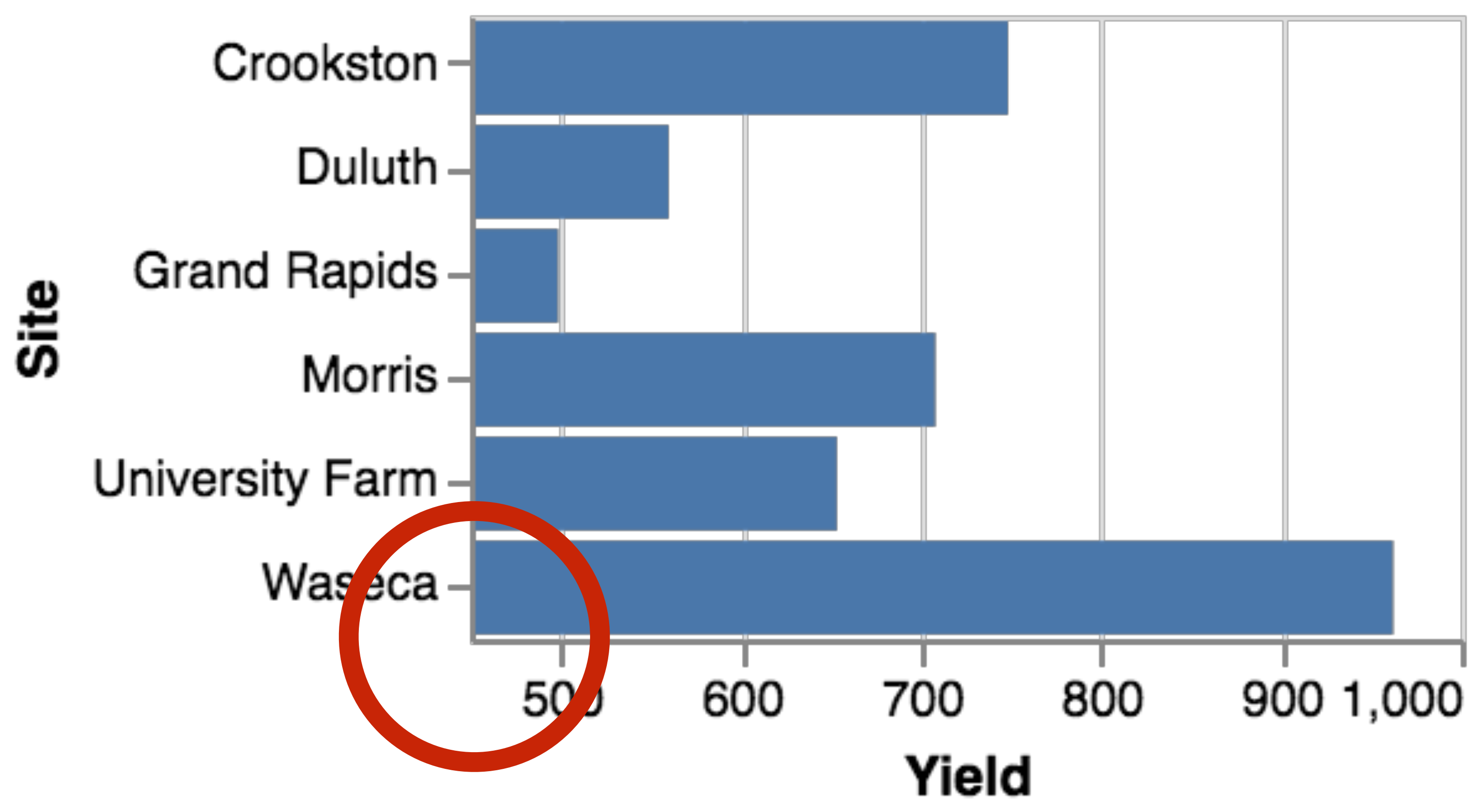


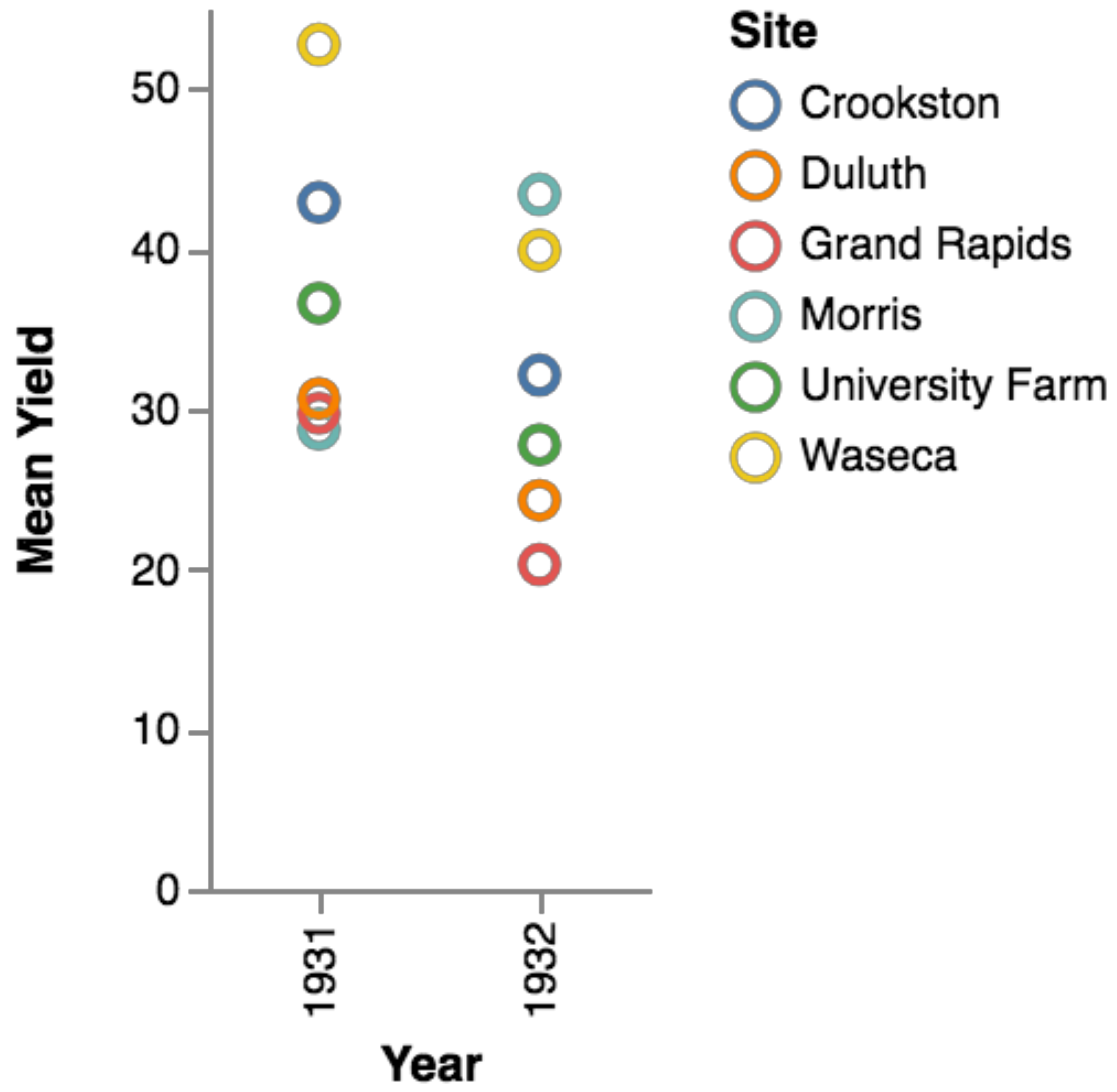
?



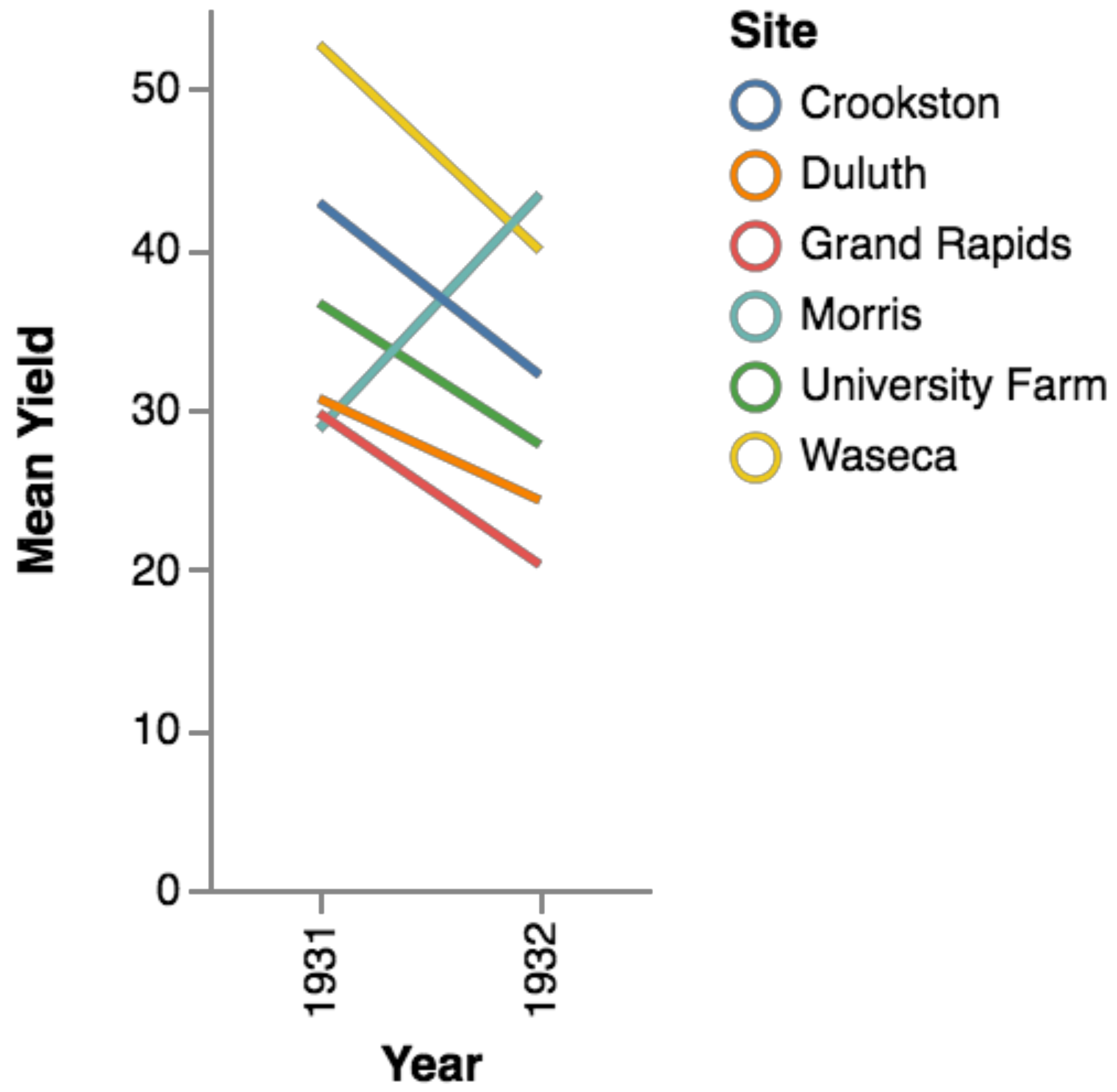

Site
Crookston
Duluth
Grand Rapids
Morris
University Farm
Waseca
0
200
400
600
800
1,000
Yield

>


Site
Crookston
Duluth
Grand Rapids
Morris
University Farm
Waseca
500
600
700
800
900
1,000
Yield


Site
Crookston
Duluth
Grand Rapids
Morris
University Farm
Waseca
Mean Yield
50
40
30
20
10
0
1931
1932
Year

<


Site
Crookston
Duluth
Grand Rapids
Morris
University Farm
Waseca
Mean Yield
50
40
30
20
10
0
1931
1932
Year

Automated Visualization Design

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

💯

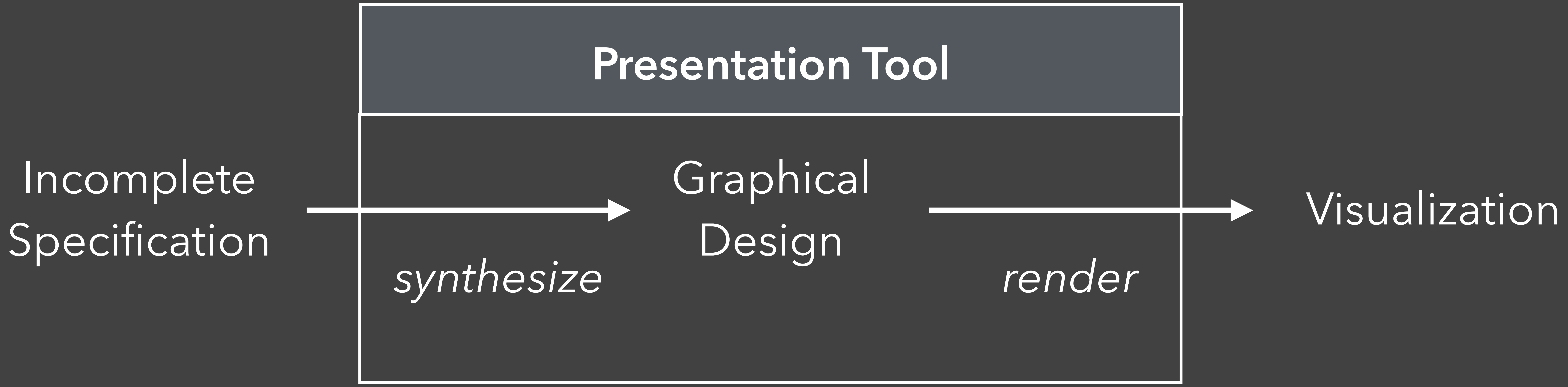
different metrics and only inferable goals

👩🏽‍💻 + 🤖

humans and machines work together

Automated Visualization Design

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



APT	**SAGE**	**BOZ**	**ShowMe**	**VizDeck**	**SeeDB**	**Voyager (2)**
Mackinlay	*Mittal et al.*	*Casner*	*Mackinlay et al.*	*Key et al.*	*Vartak et al.*	*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.

Reusable and Extensible **Knowledge Base**

Automated **Reasoning**

Outline

Modeling Visualization Design

Applying Visualization Design

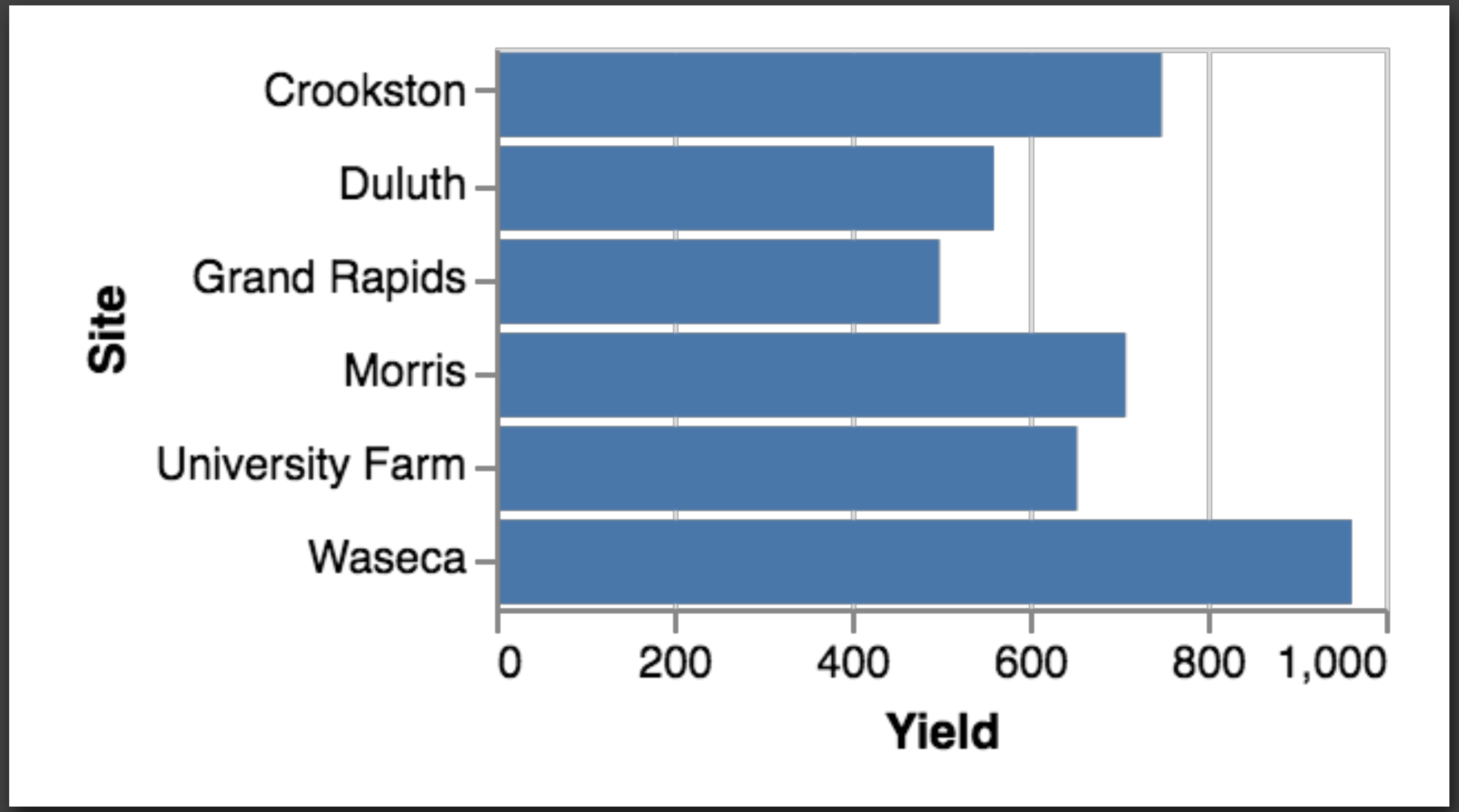
Learning Visualization Design

Modeling Visualization Design in Draco

Visualizations → Logical Facts

Design Knowledge → Constraints

Visualizations → Logical Facts



$$\text{data} = \text{barley}$$

$$\text{mark} = \text{bar}$$

$$\text{encoding1} = \left\{ \begin{array}{ll} \text{channel} & = \text{x} \\ \text{field} & = \text{yield} \\ \text{type} & = \text{continuous} \\ \text{aggregate} & = \text{sum} \\ \text{zero} & = \text{T} \end{array} \right\}$$

$$\text{encoding2} = \left\{ \begin{array}{ll} \text{channel} & = \text{y} \\ \text{field} & = \text{site} \\ \text{type} & = \text{categorical} \end{array} \right\}$$

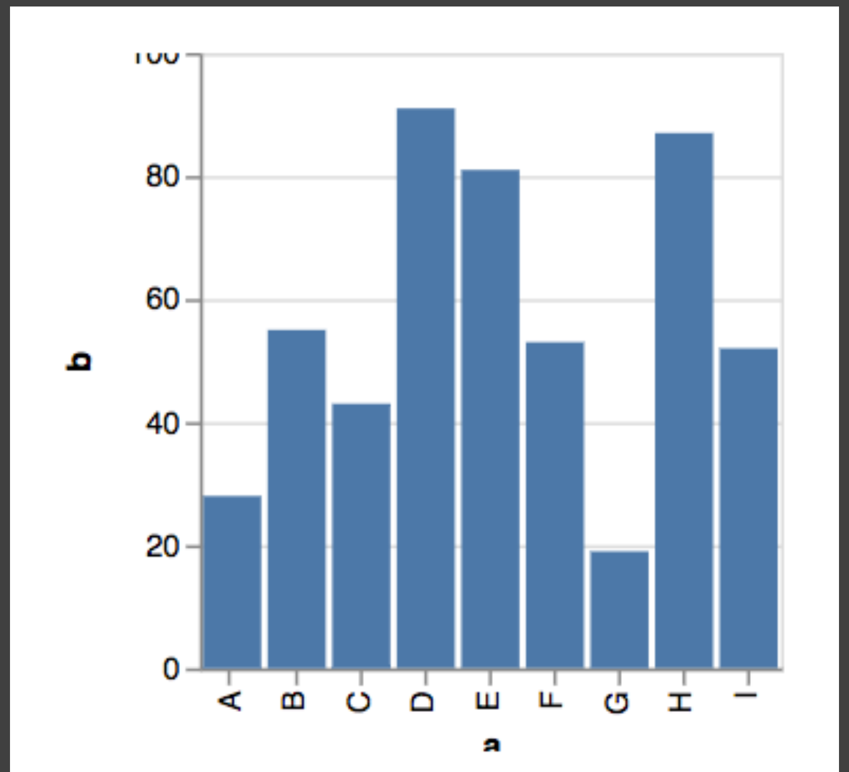
Design Knowledge → Constraints

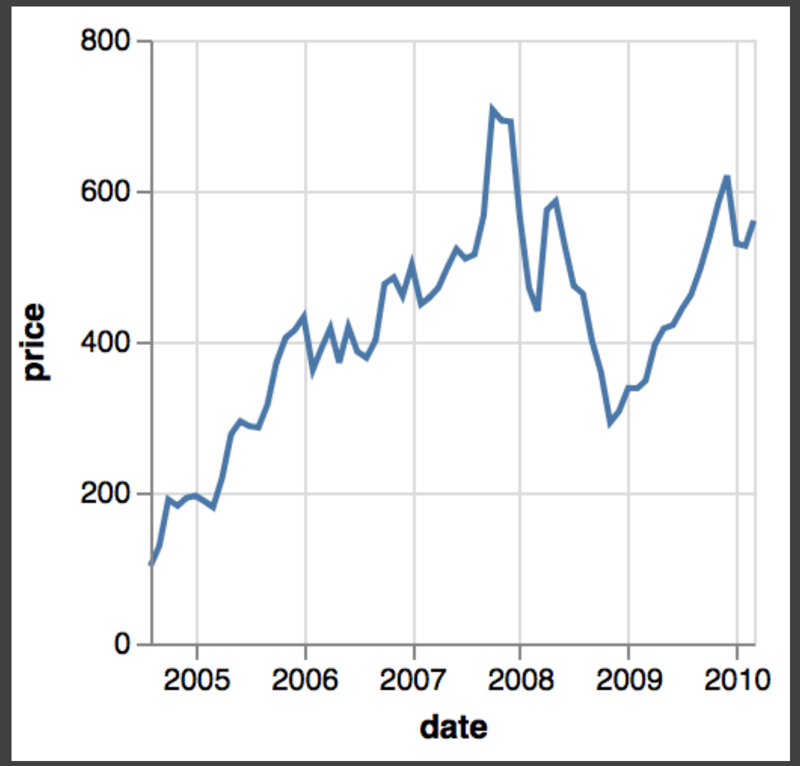
Visualizations → Logical Facts

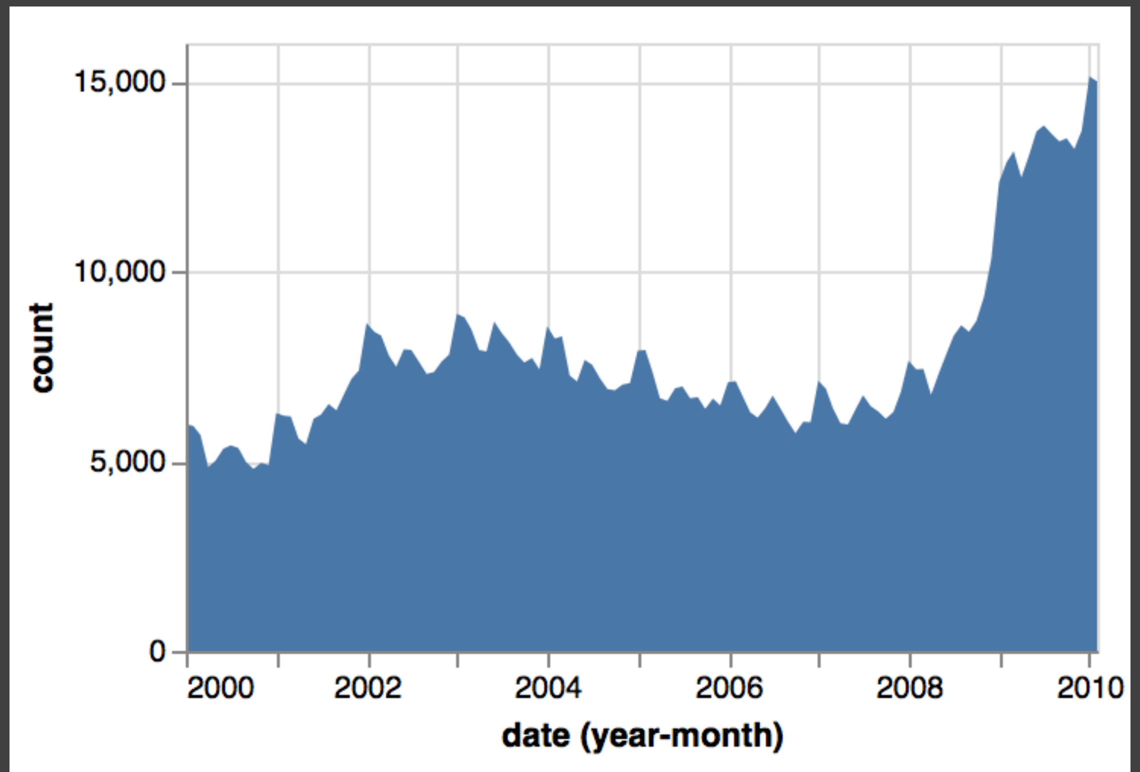
Design Knowledge → Constraints

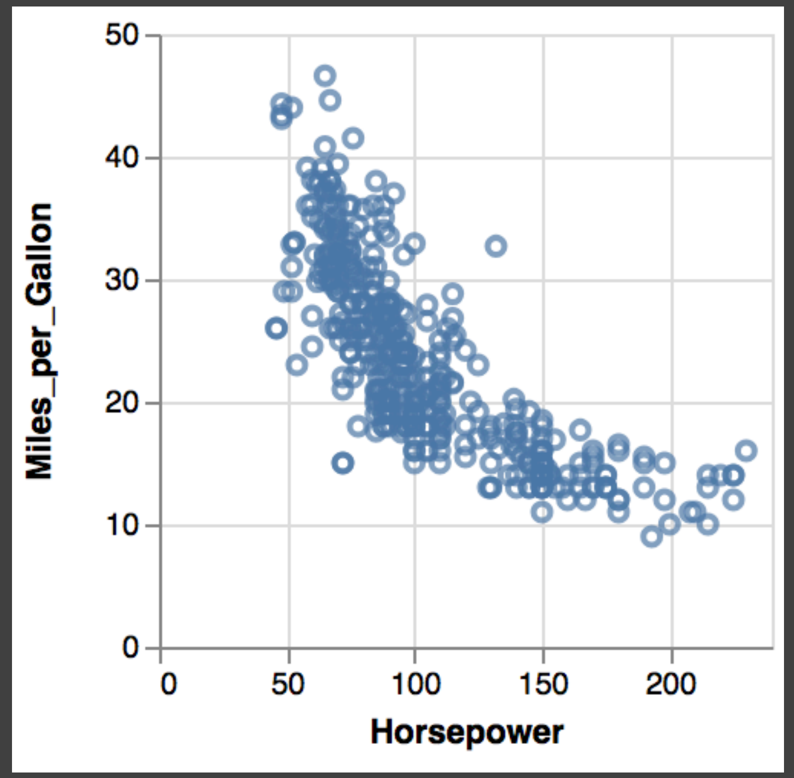
1. Domain of Attributes

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









mark ∈ {bar, line, area, point}

Visualizations → Logical Facts

Design Knowledge → Constraints

1. Domain of Attributes

mark $\in$ {bar, line, area, point}

channel $\in$ {x, y, color, text, opacity, shape}

field $\in$ {site, variety, yield, year, age}

type $\in$ {categorical, continuous}

aggregate $\in$ {sum, count, mean, median}

zero $\in \{ \perp , \top \}$

bin $\in \mathbb{N}$

…

Visualizations → Logical Facts

Design Knowledge → Constraints

2. Integrity Constraints

"Only ***continuous*** fields can be ***aggregated***"

$$\forall e \in \text{Encodings} : e\,.\,\text{aggregate} \Rightarrow e\,.\,\text{type} = \text{continuous}$$

Design Knowledge → Constraints

2. Integrity Constraints

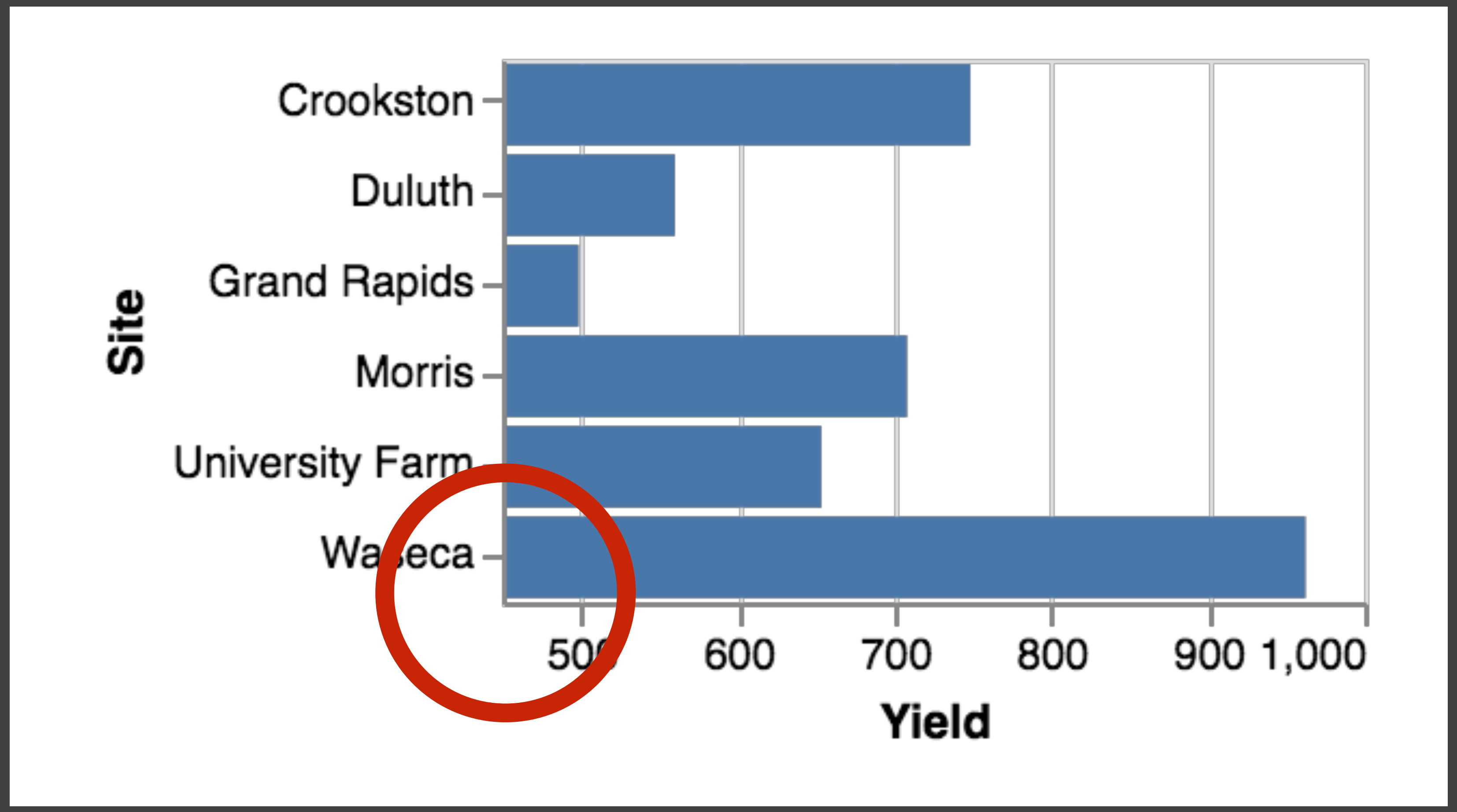
$$\forall e \in \text{Encodings} : e\,.\,\text{aggregate} \Rightarrow e\,.\,\text{type} = \text{continuous}$$
$$(\exists e \in \text{Encodings} : e\,.\,\text{channel} = \text{shape}) \Rightarrow \text{mark} = \text{point}$$
$$\text{mark} = \text{bar} \Rightarrow \exists e \in \text{Encodings} : (e\,.\,\text{channel} = \text{x} \vee e\,.\,\text{channel} = \text{y})$$
$$\ldots$$

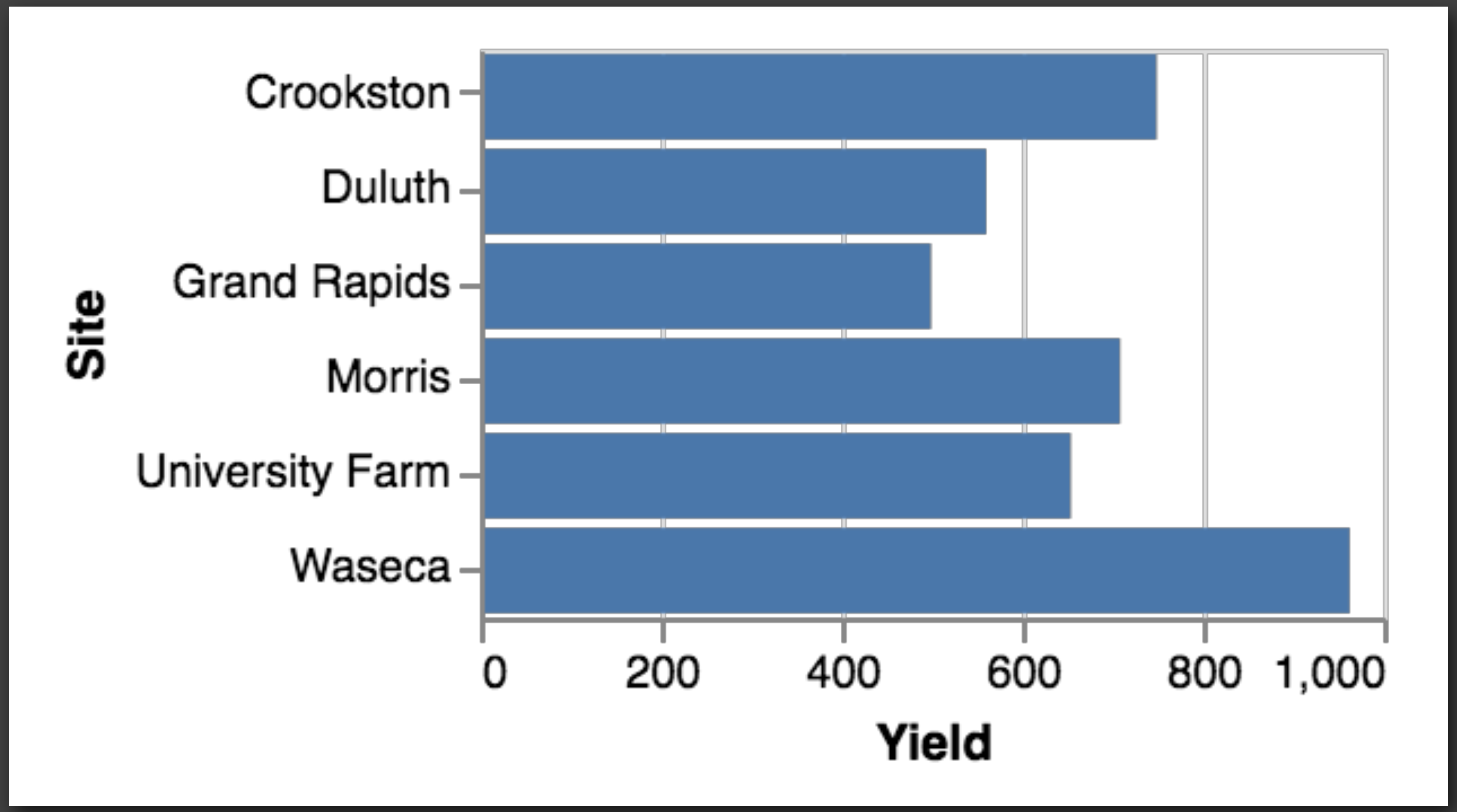
Visualizations → Logical Facts

Design Knowledge → Constraints

3. Preferences

"Prefer to include ***zero***."





Cost if soft constraint is violated

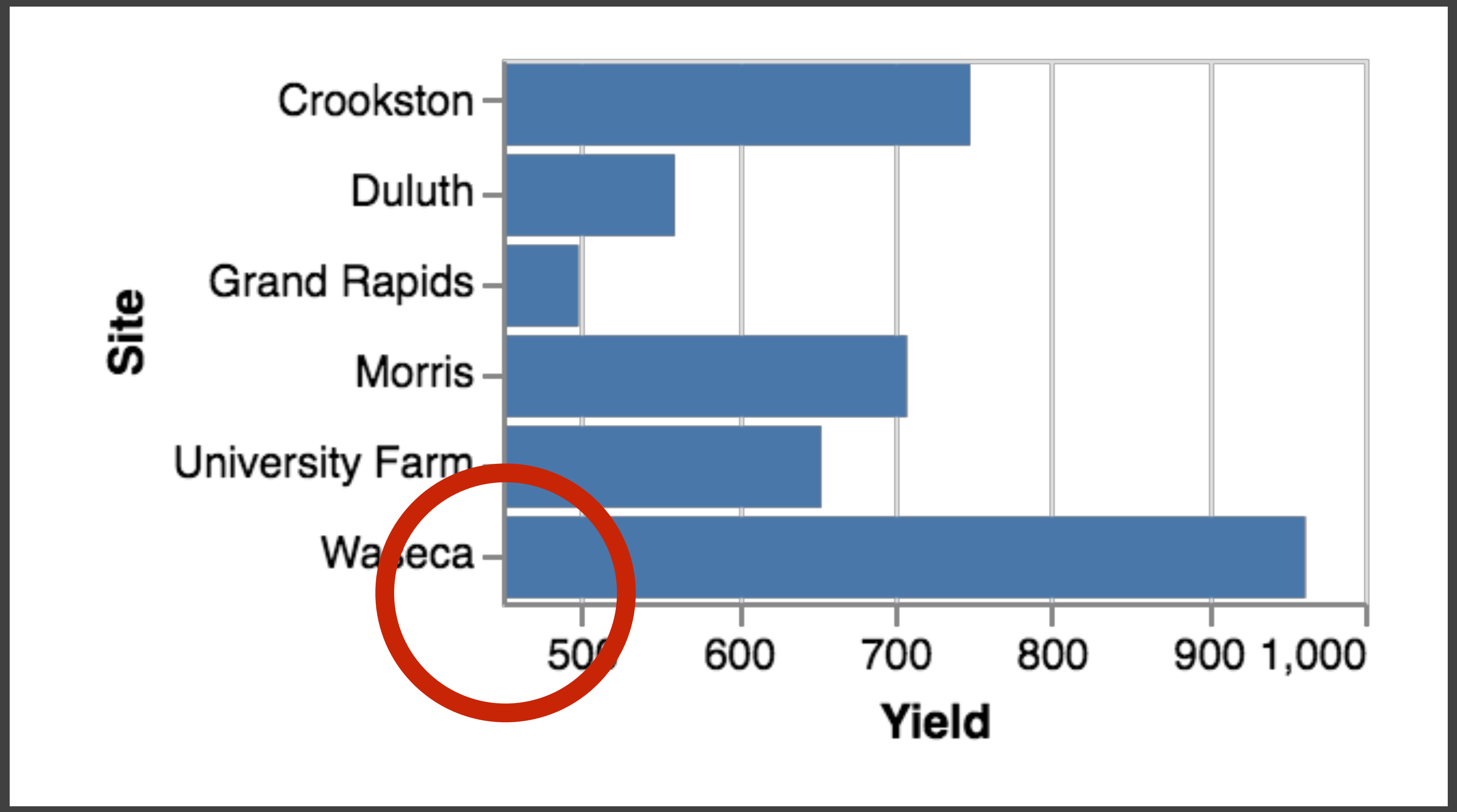
$$\mathbf{5}\ \forall e \in \text{Encodings} : e.\text{zero} = \text{T}$$

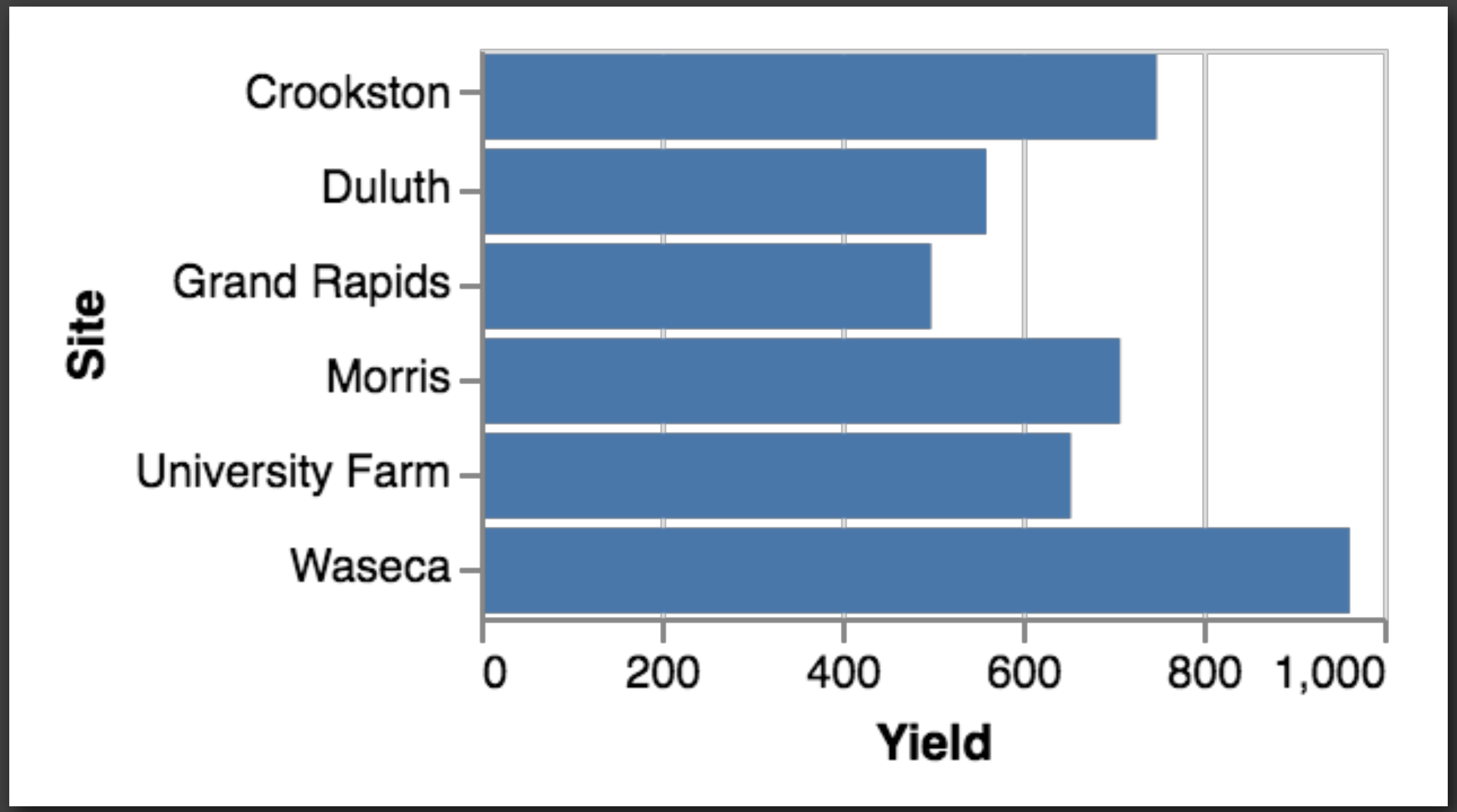
Visualizations → Logical Facts

Design Knowledge → Constraints

3. Preferences

"Prefer to include ***zero*** for **continuous** fields."





Cost if soft constraint is violated

$\mathbf{5}\,\forall e \in \text{Encodings} : e.\text{type} = \text{continuous} \Rightarrow e.\text{zero} = \text{T}$

Visualizations → Logical Facts

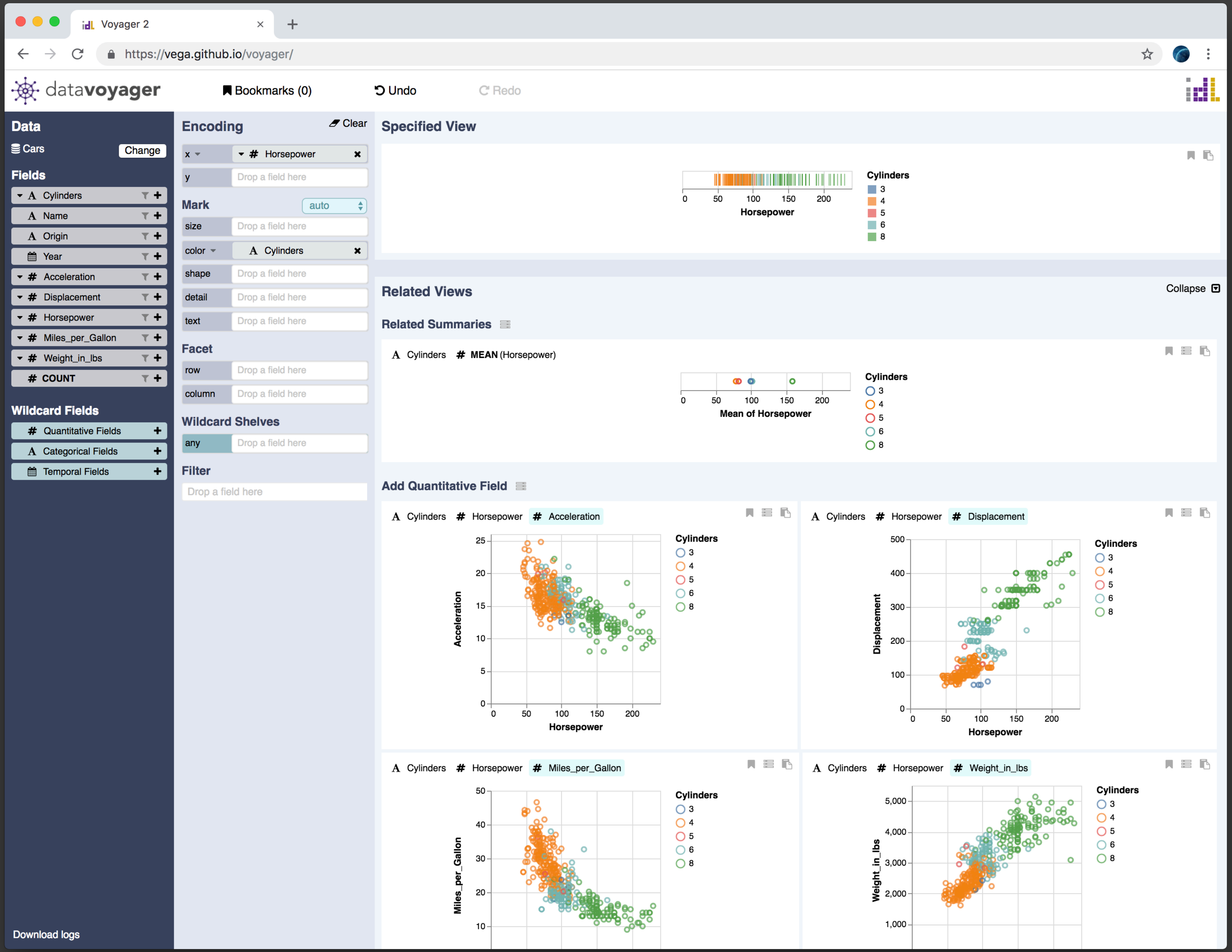
Design Knowledge → Constraints

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

Automated Visualization Design

Verifying Designs

Visualization Recommendation

Autocomplete

Comparing Visualization Models

Enumerating Design Space

Constrain Models

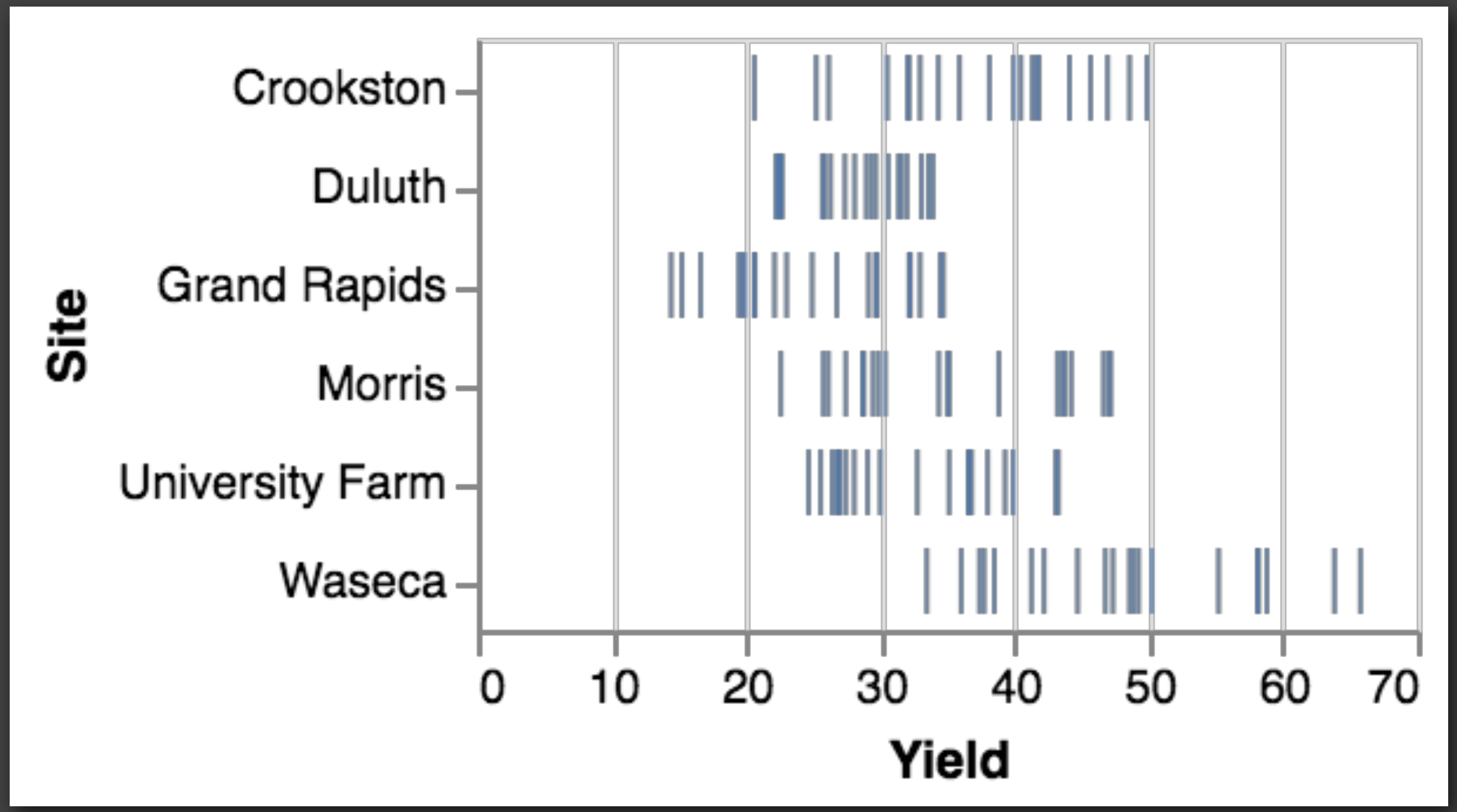
Synthesis

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**."



$$\text{data} = \text{barley} \land$$
$$\exists e : e.\text{field} = \text{site} \land$$
$$\exists e : e.\text{field} = \text{yield}$$

1. Domain of Attributes
2. Integrity Constraints
3. Preferences

Clingo solver

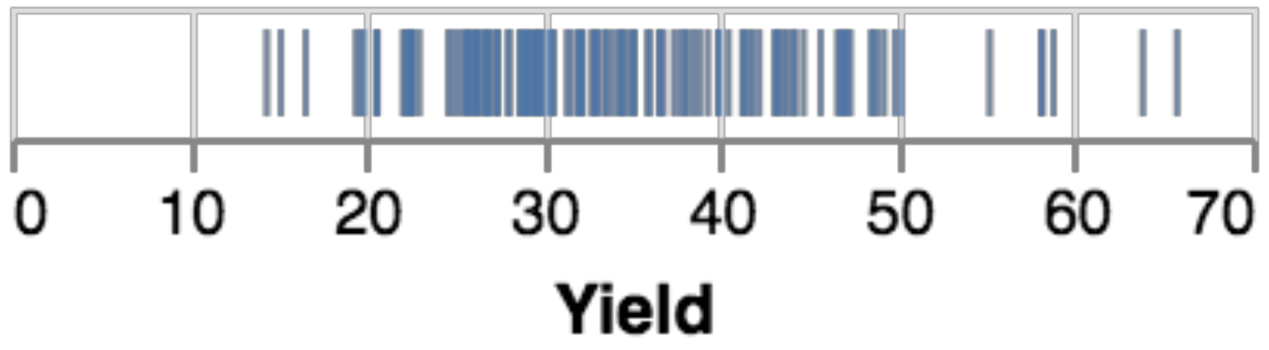
data = barley

mark = tick

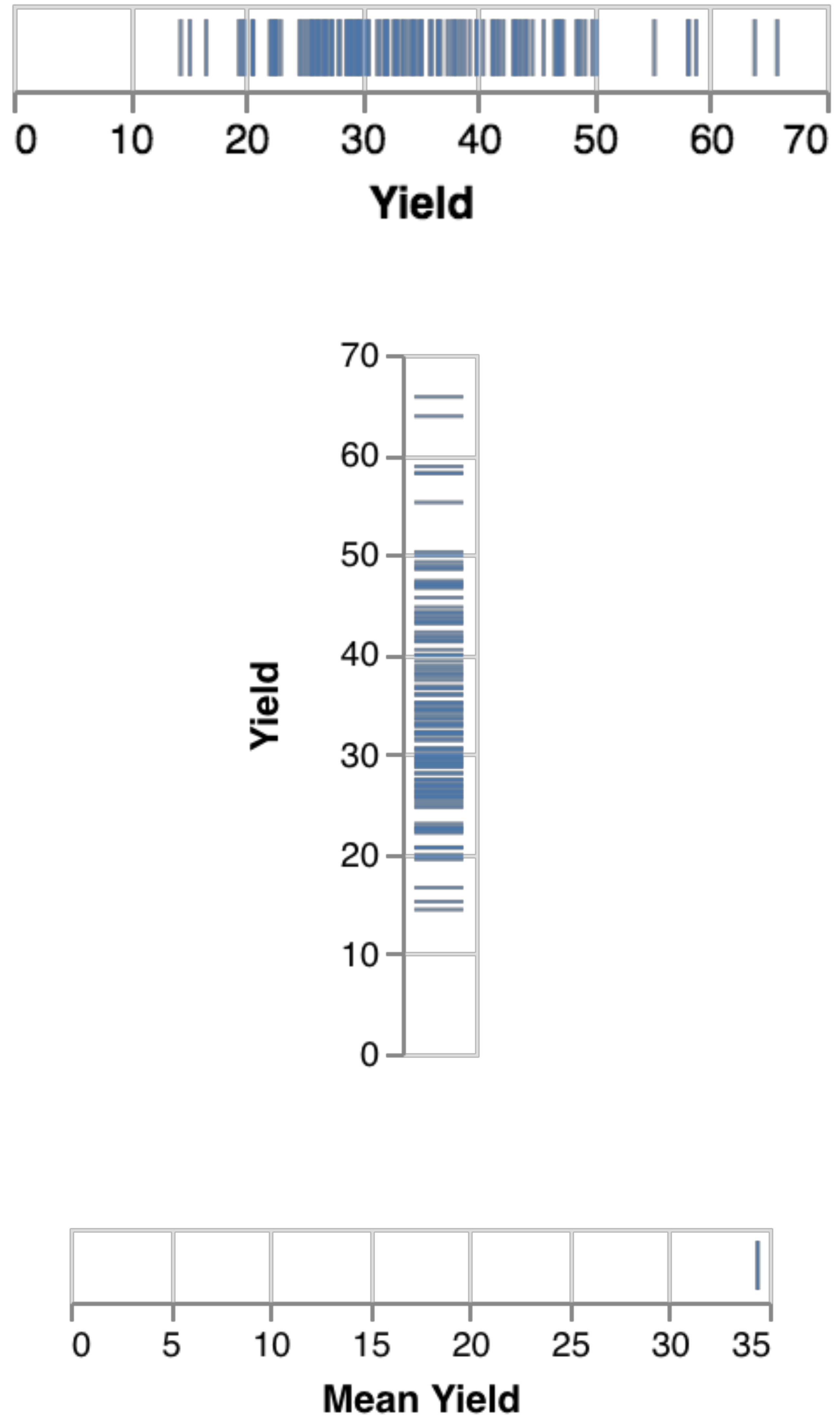
$$\text{encoding1} = \left\{ \begin{array}{ll} \text{channel} & = x \\ \text{field} & = \text{yield} \\ \text{type} & = \text{quantitative} \\ \text{zero} & = \text{T} \end{array} \right\}$$

$$\text{encoding2} = \left\{ \begin{array}{ll} \text{channel} & = y \\ \text{field} & = \text{site} \\ \text{type} & = \text{categorical} \end{array} \right\}$$

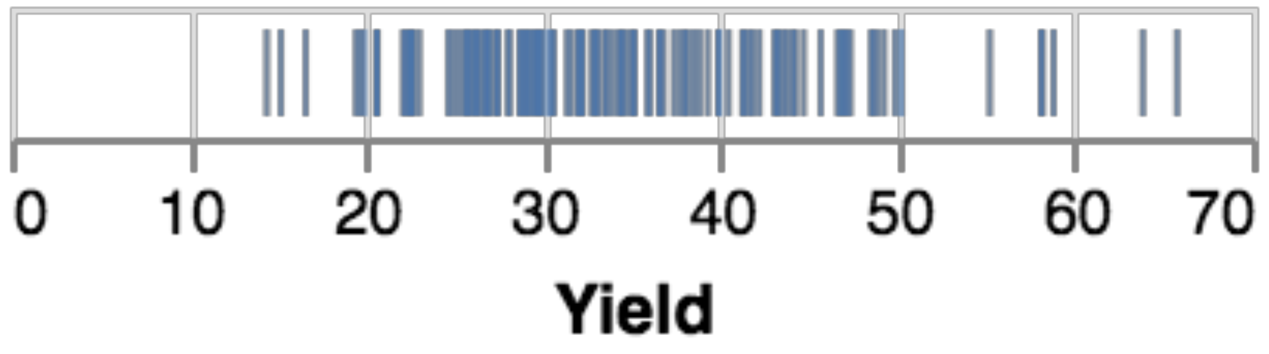
$\text{data} = \text{barley} \wedge$
$\exists e : e.\text{field} = \text{yield}$



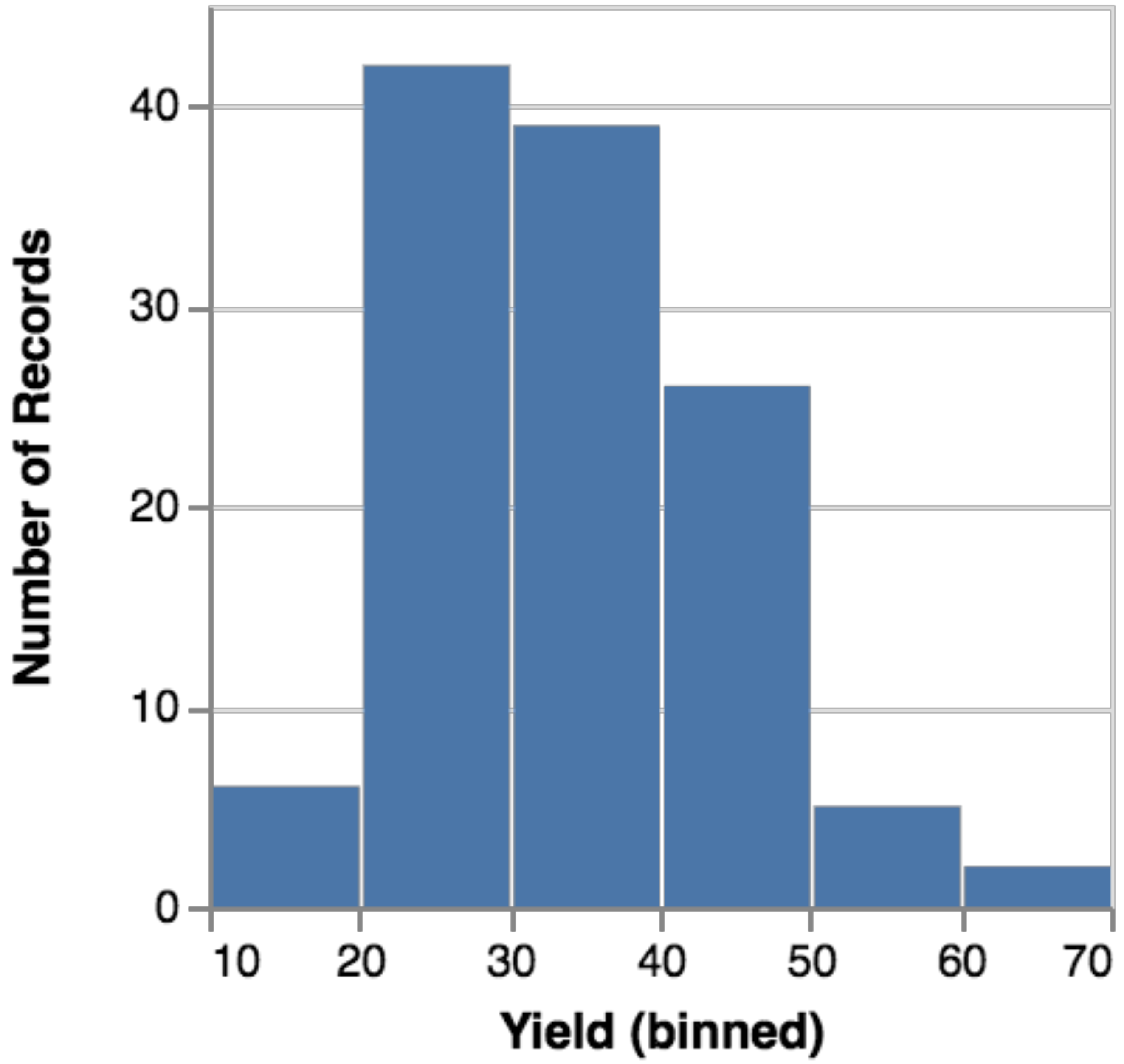
$$\text{data} = \text{barley} \land$$
$$\exists e : e\,.\,\text{field} = \text{yield}$$



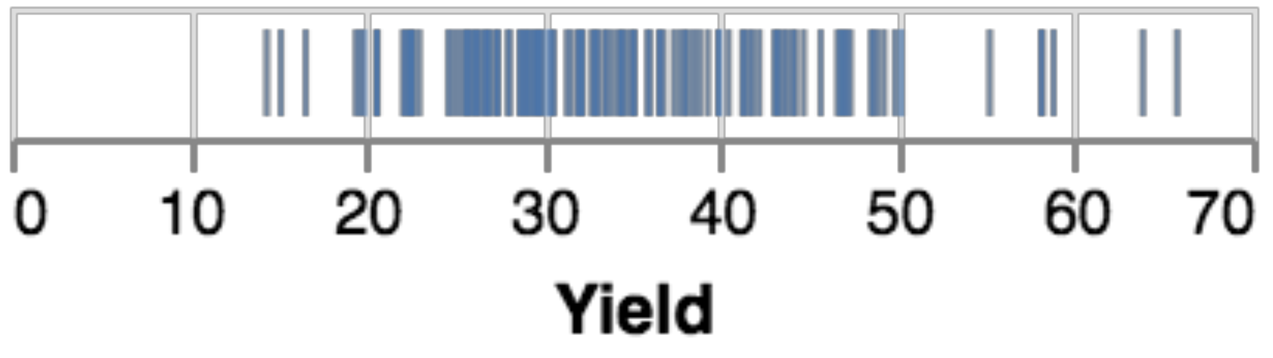
$$\text{data} = \text{barley} \wedge$$
$$\exists e : e.\text{field} = \text{yield}$$



$$\text{data} = \text{barley} \wedge$$
$$\exists e : e.\text{field} = \text{yield} \wedge e.\text{bin}$$

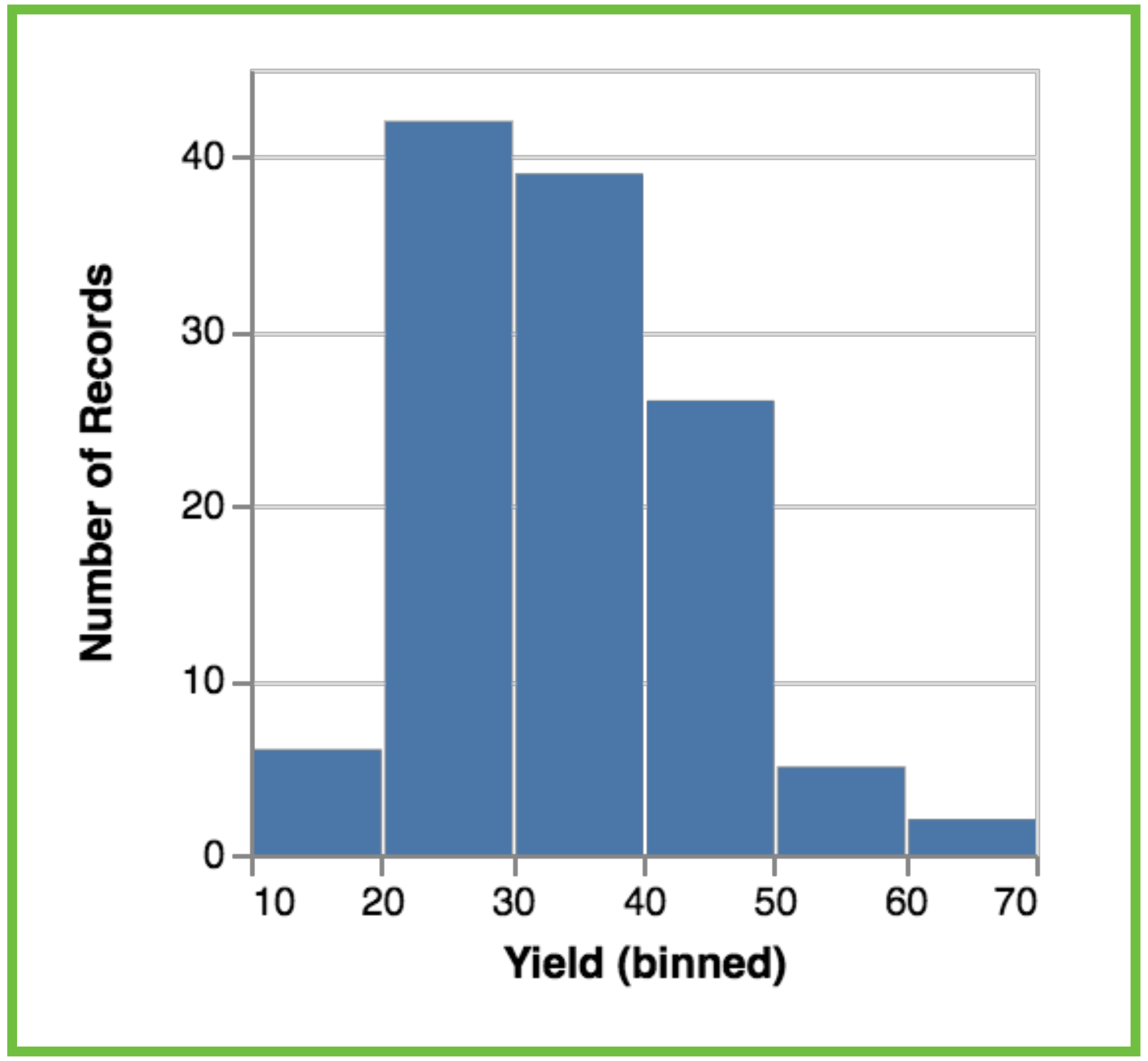


$\text{data} = \text{barley} \wedge$
$\exists e : e.\text{field} = \text{yield}$



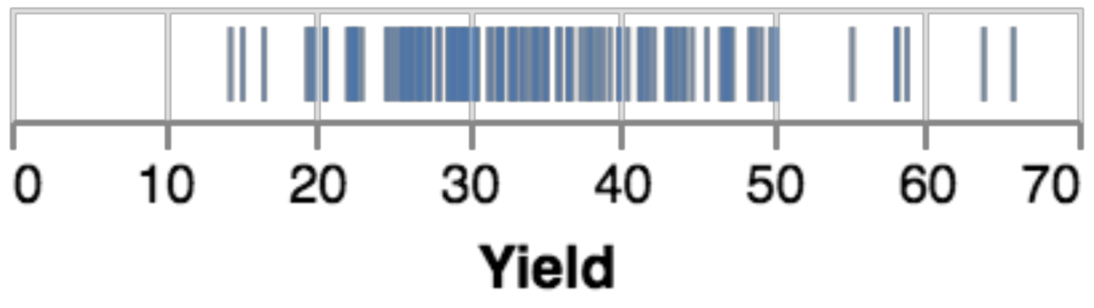
$\text{data} = \text{barley} \wedge$
$\exists e : e.\text{field} = \text{yield} \wedge e.\text{bin}$

1 not $e \in$ Encoding
1 $\forall e \in$ Encoding : e.aggregate != count
3 $\exists e : e$.continuous = T $\Rightarrow \exists e \in$ Encoding : e.aggregate

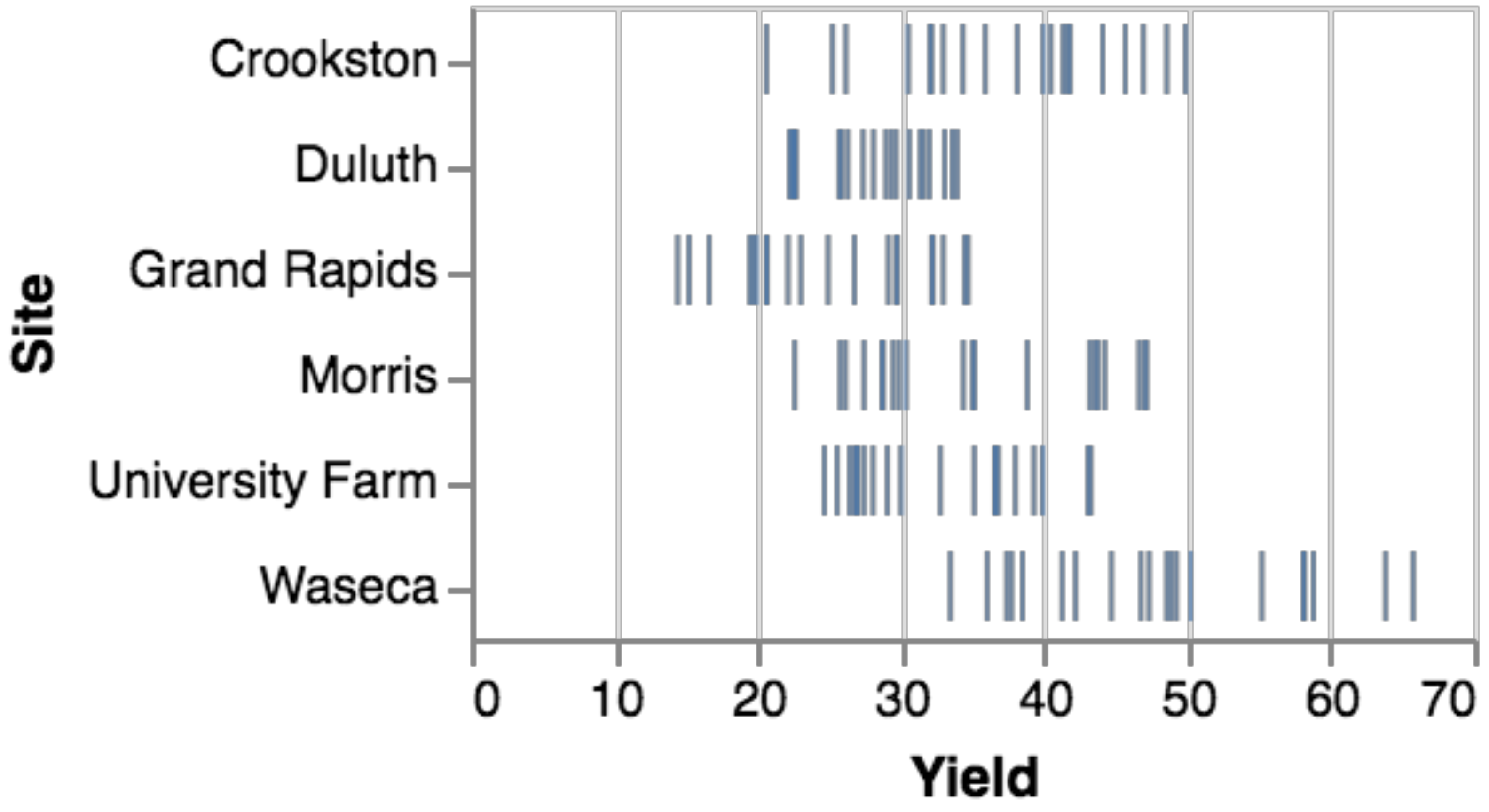


Adding count prevents overlap

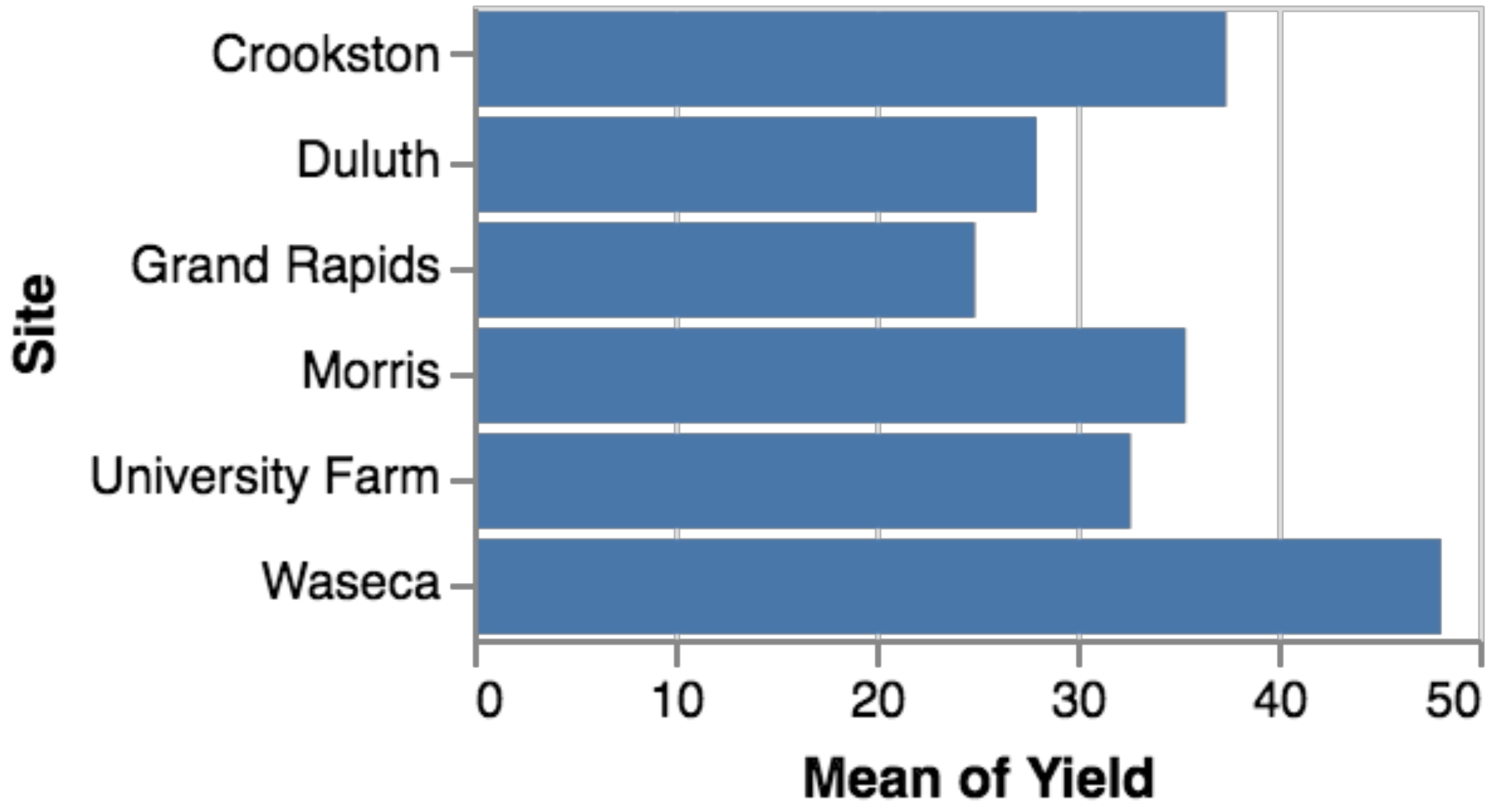
$$\text{data} = \text{barley} \wedge$$
$$\exists e : e.\text{field} = \text{yield}$$



$$\text{data} = \text{barley} \wedge$$
$$\exists e : e.\text{field} = \text{yield} \wedge$$
$$\exists e : e.\text{field} = \text{site}$$



$$\text{data} = \text{barley} \wedge$$
$$\exists e : e.\text{field} = \text{yield} \wedge$$
$$\exists e : e.\text{field} = \text{site} \wedge$$
$$\text{mark} = \text{bar}$$



$\text{data} = \text{barley} \wedge$

$\exists e : e.\text{field} = \text{yield}$

1 mark != tick

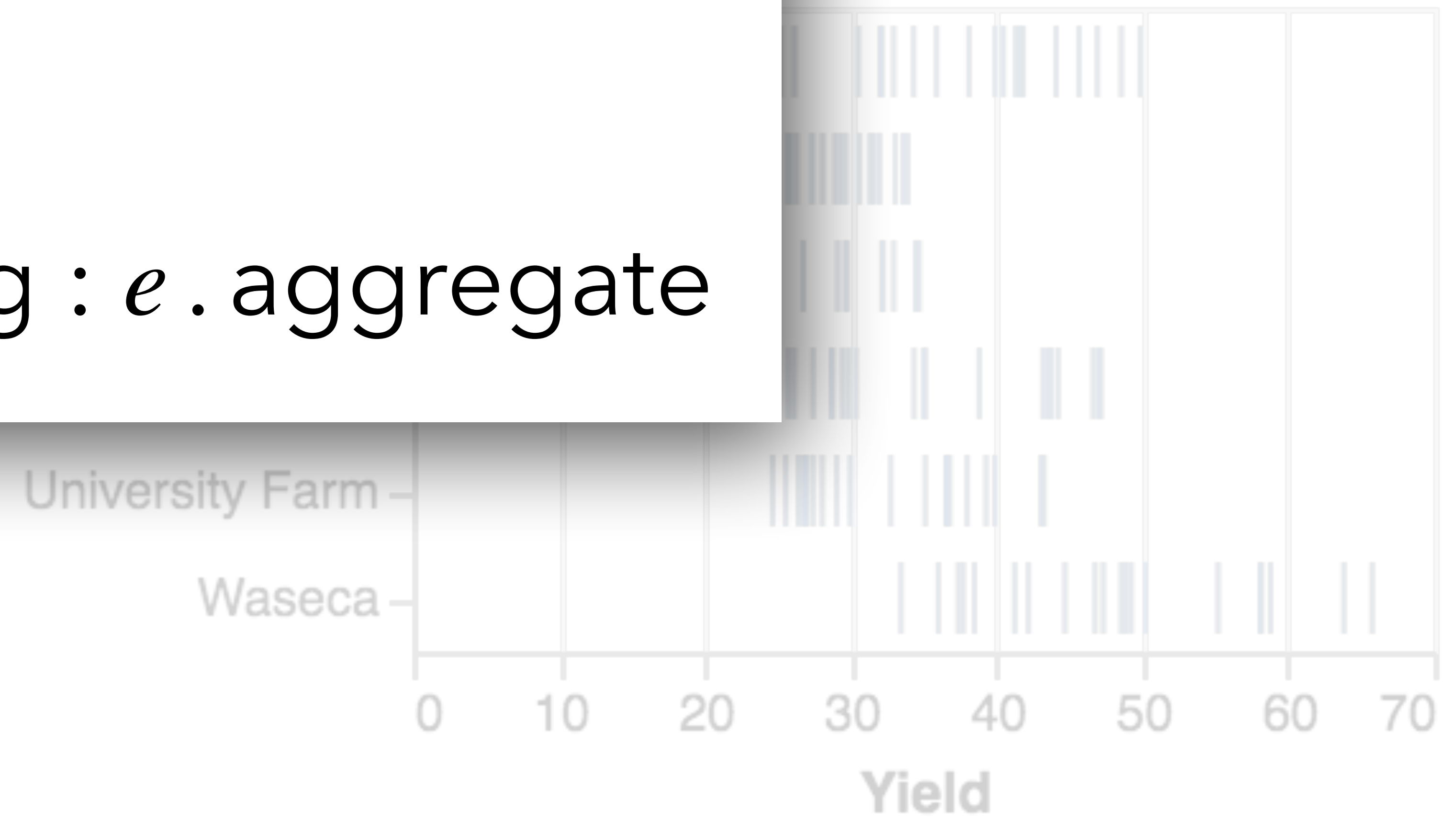
2 mark != bar

3 $\text{mark} = \text{bar} \Rightarrow \exists e \in \text{Encoding} : e.\text{aggregate}$

$\text{data} = \text{barley} \wedge$

$\exists e : e.\text{field} = \text{y}$

$\exists e : e.\text{field} = \text{site}$

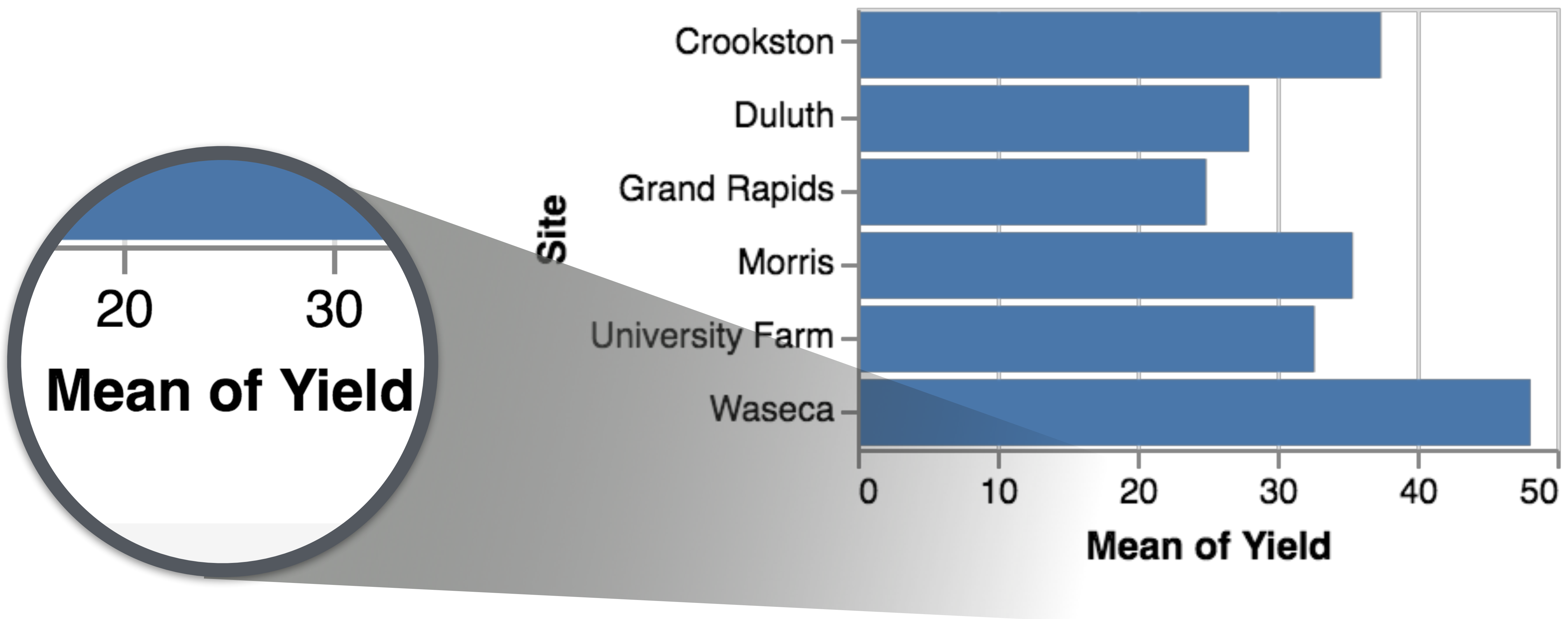


$\text{data} = \text{barley} \wedge$

$\exists e : e.\text{field} = \text{yield} \wedge$

$\exists e : e.\text{field} = \text{site} \wedge$

$\text{mark} = \text{bar}$



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

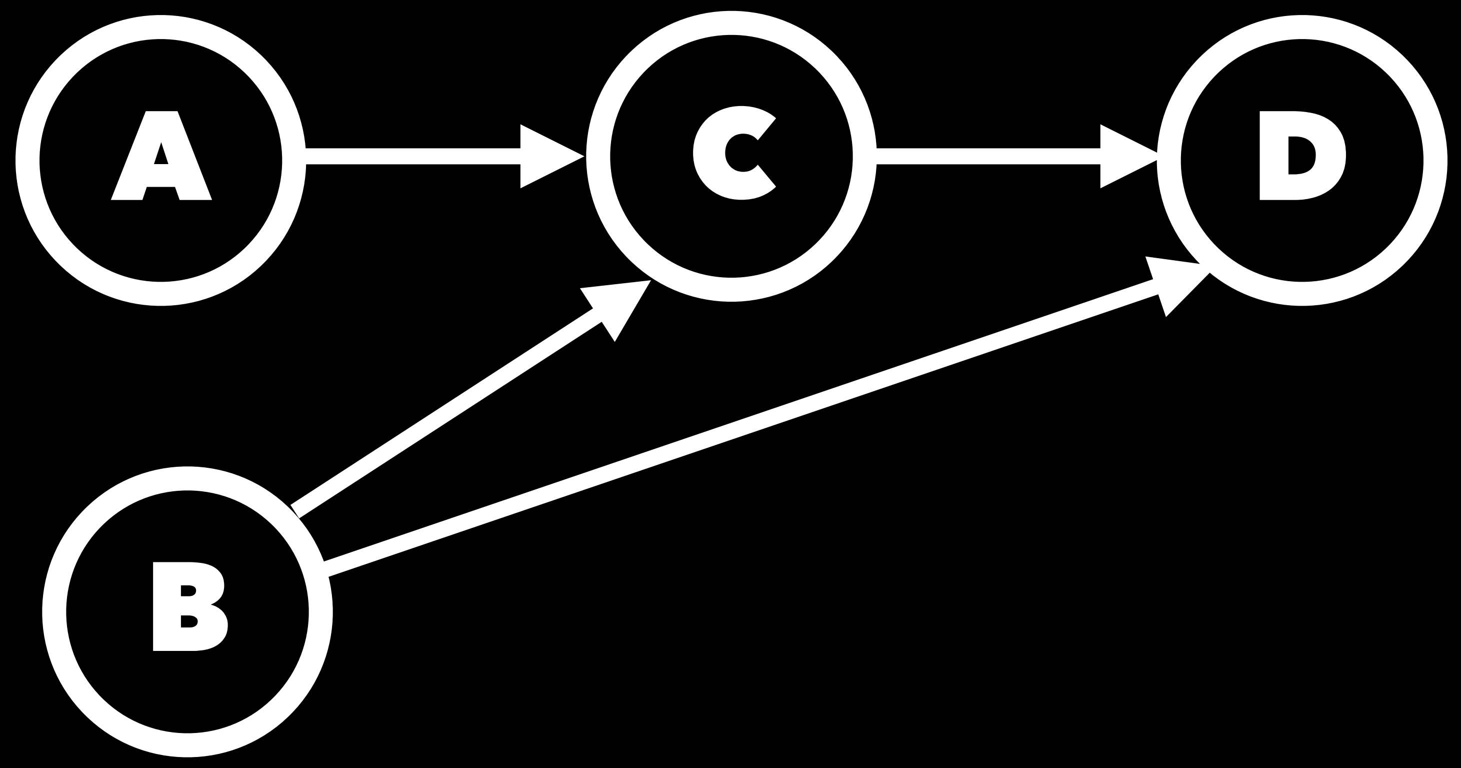
70 Hard Constraints

110 Soft Constraints

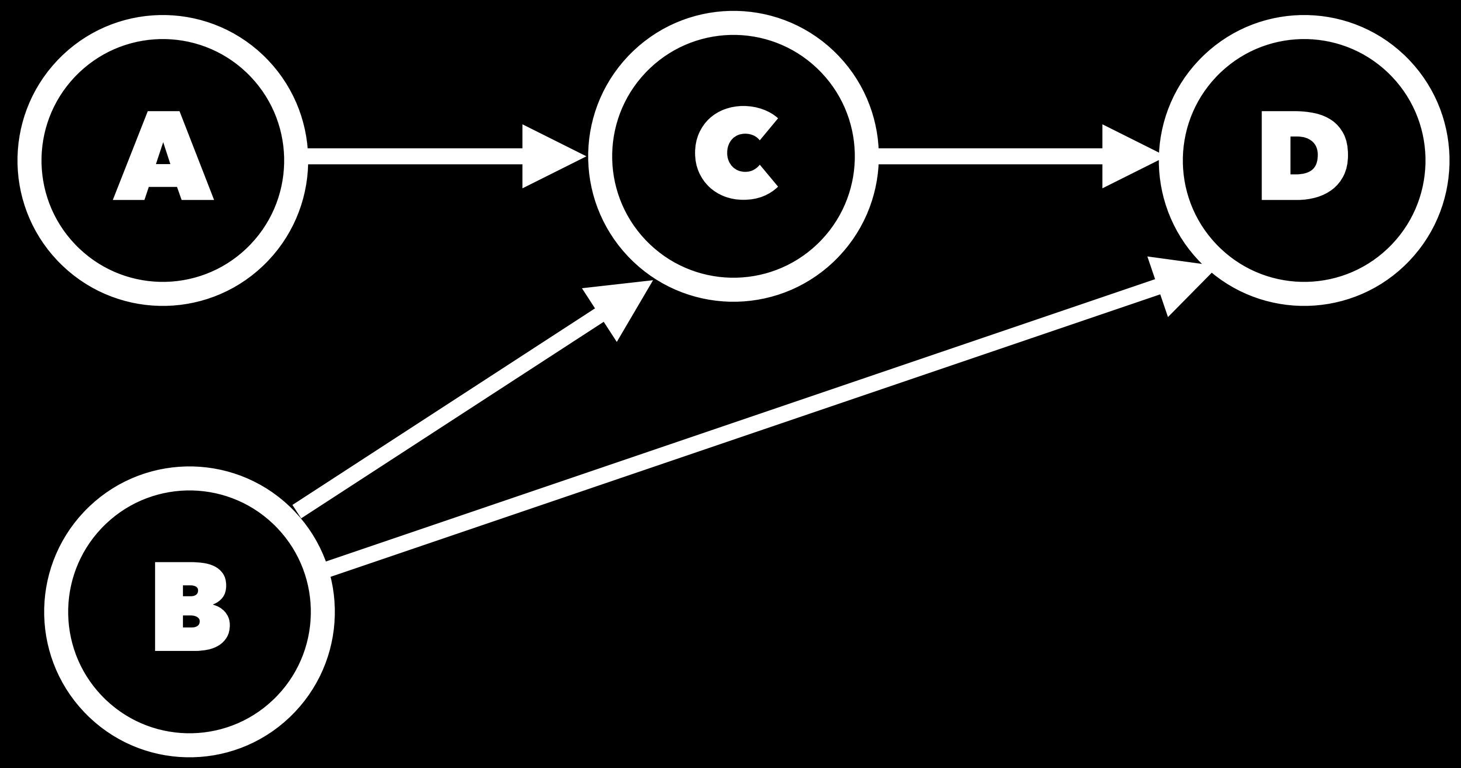
Wait... 🤔

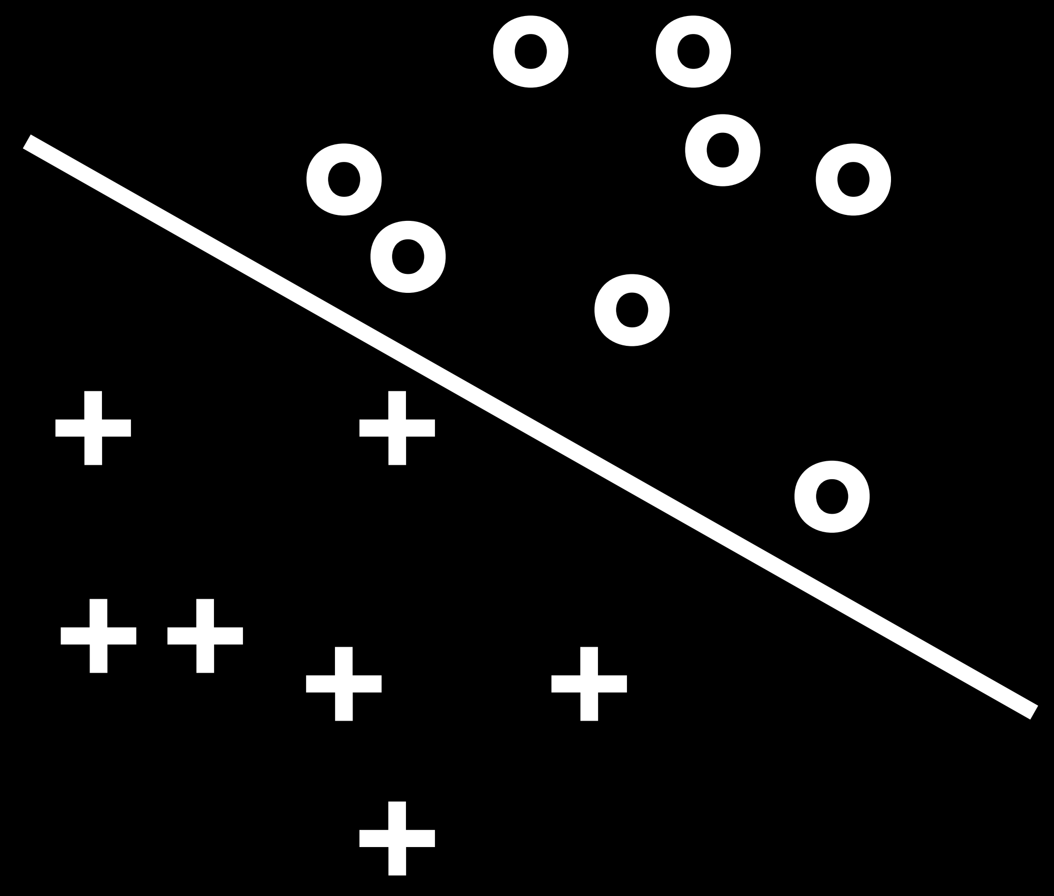
How to weight rules?

Classical AI



Classical AI + Machine Learning





Outline

Modeling Visualization Design

Applying Visualization Design

Learning Visualization Design

Learning Visualization Design in Draco

Features

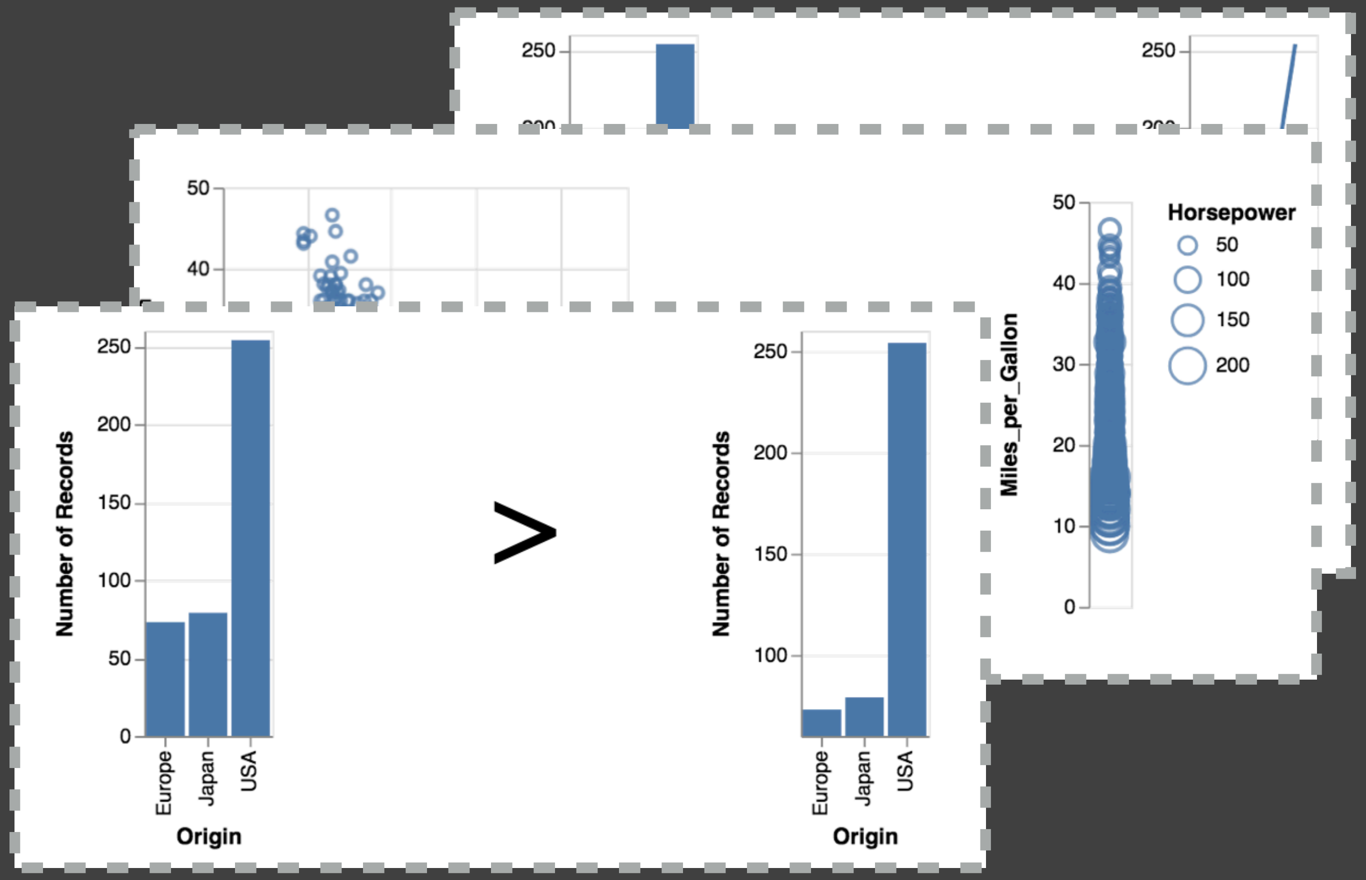
Violations of
Soft Constraints

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

v_i : the number of violations of rule i.

Training Data

Pairs of Ranked
Visualizations

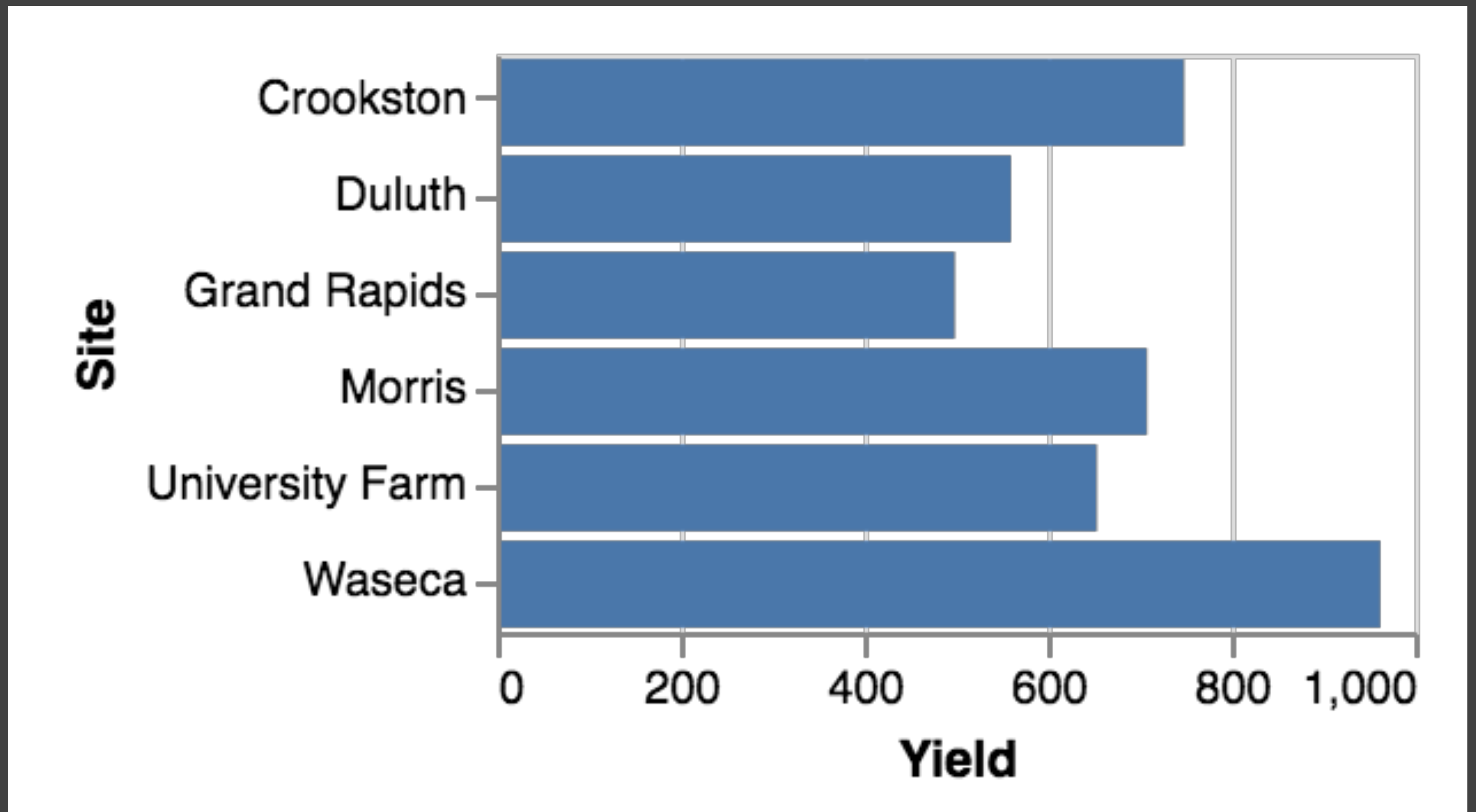


Learning Algorithm

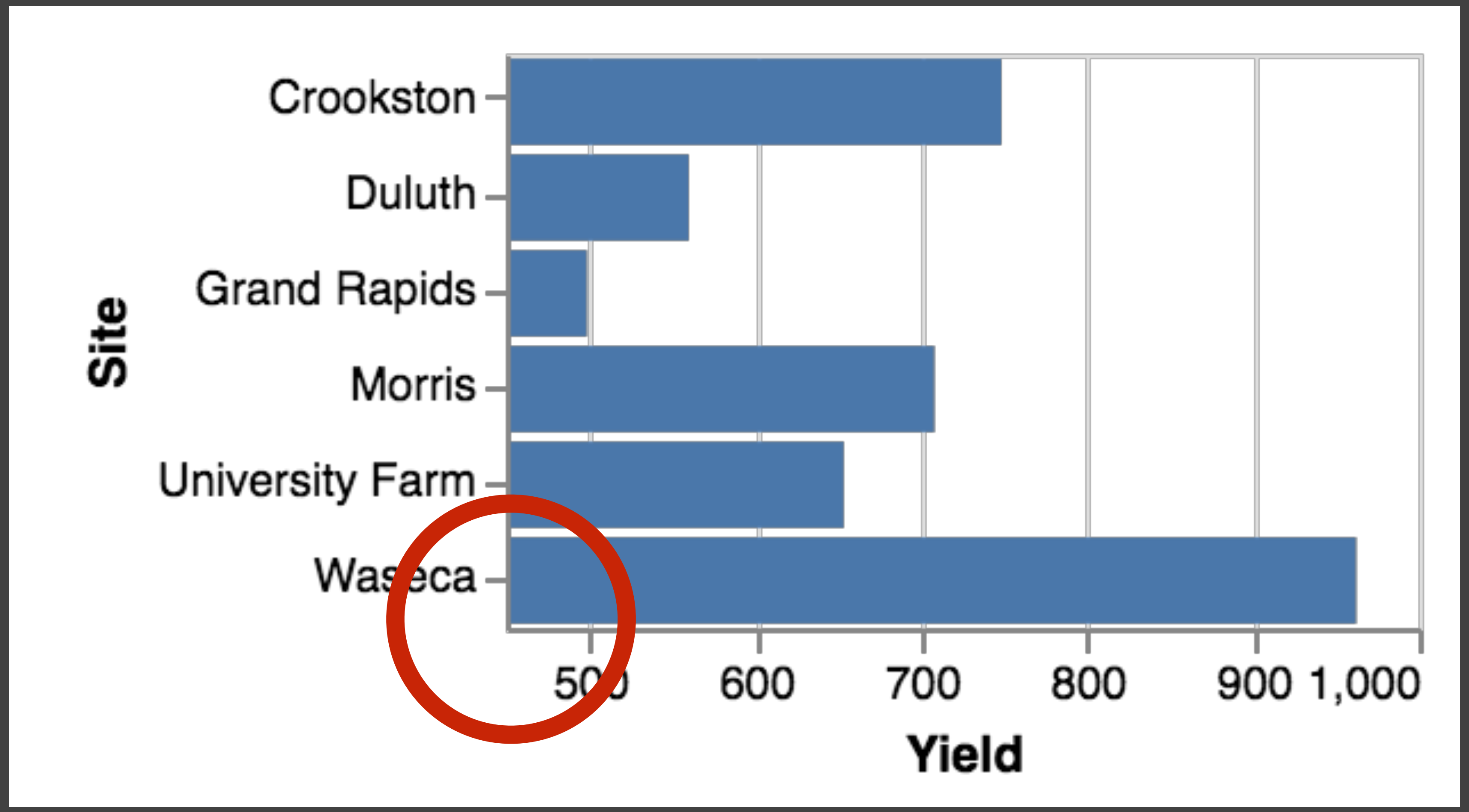
Learning to Rank
with Linear SVM

$$L = \frac{1}{n} \sum_{i=1}^{k} \max\left(0, 1 - y_i \mathbf{w}^T (\mathbf{x}_{i1} - \mathbf{x}_{i2})\right)$$
$$+\lambda \|w\|_2$$
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} L$$

Violations of Soft Constraints as Features



>



Violates:

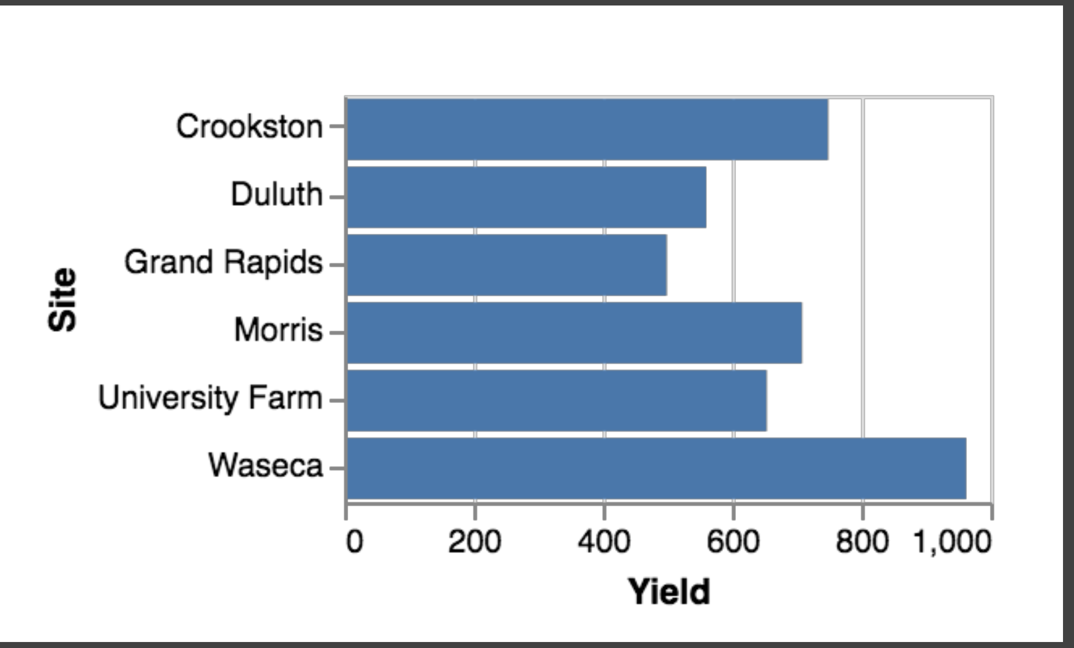
$\mathbf{w}\,\forall e \in \text{Encodings}:$

$e.\text{type} = \text{continuous} \Rightarrow$

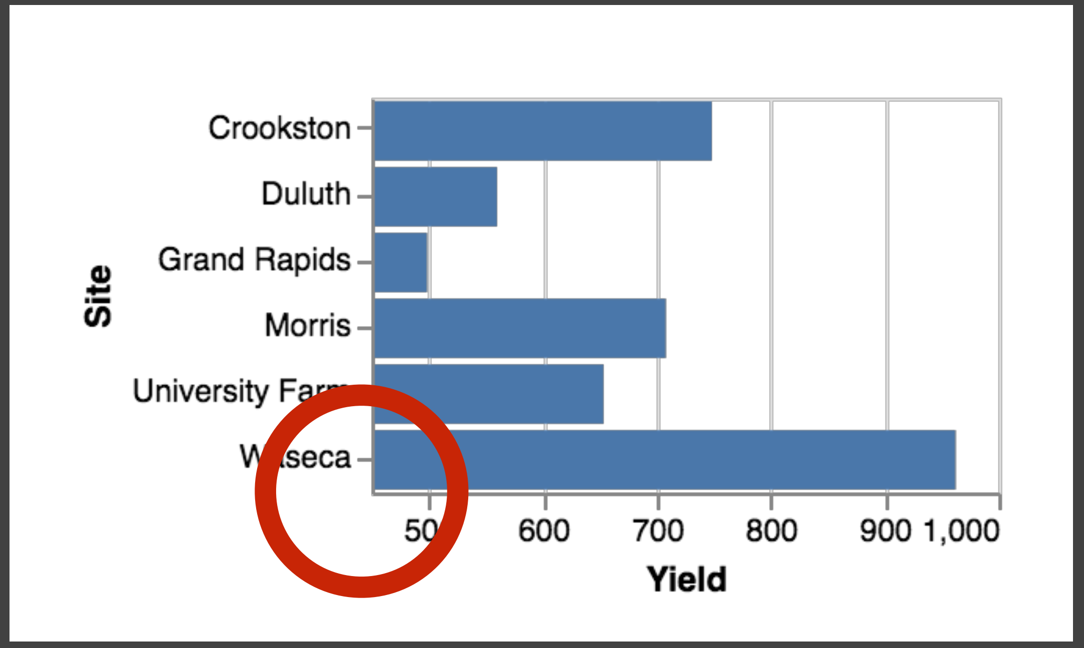
$e.\text{zero} = \text{T}$

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

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

v_i : the number of violations of constraint i

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

w is the weight vector of the soft constraints

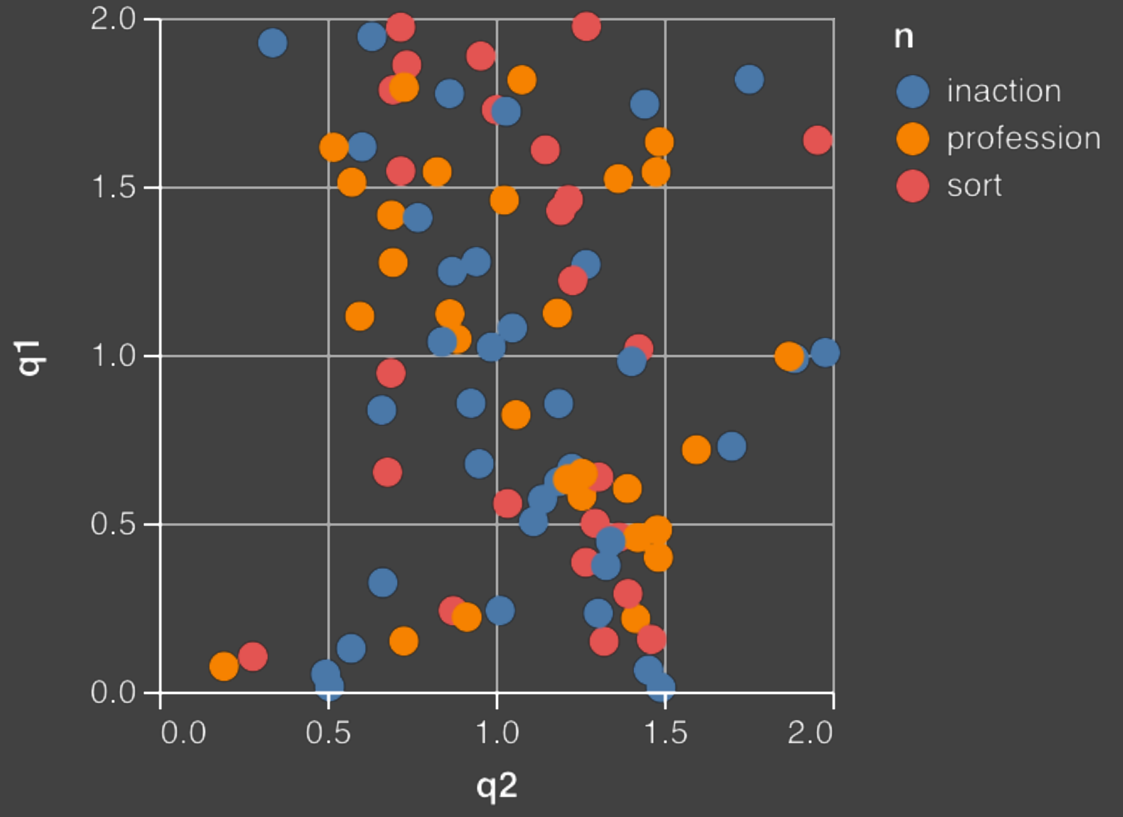
Example: Draco-Learn

Draco-CQL

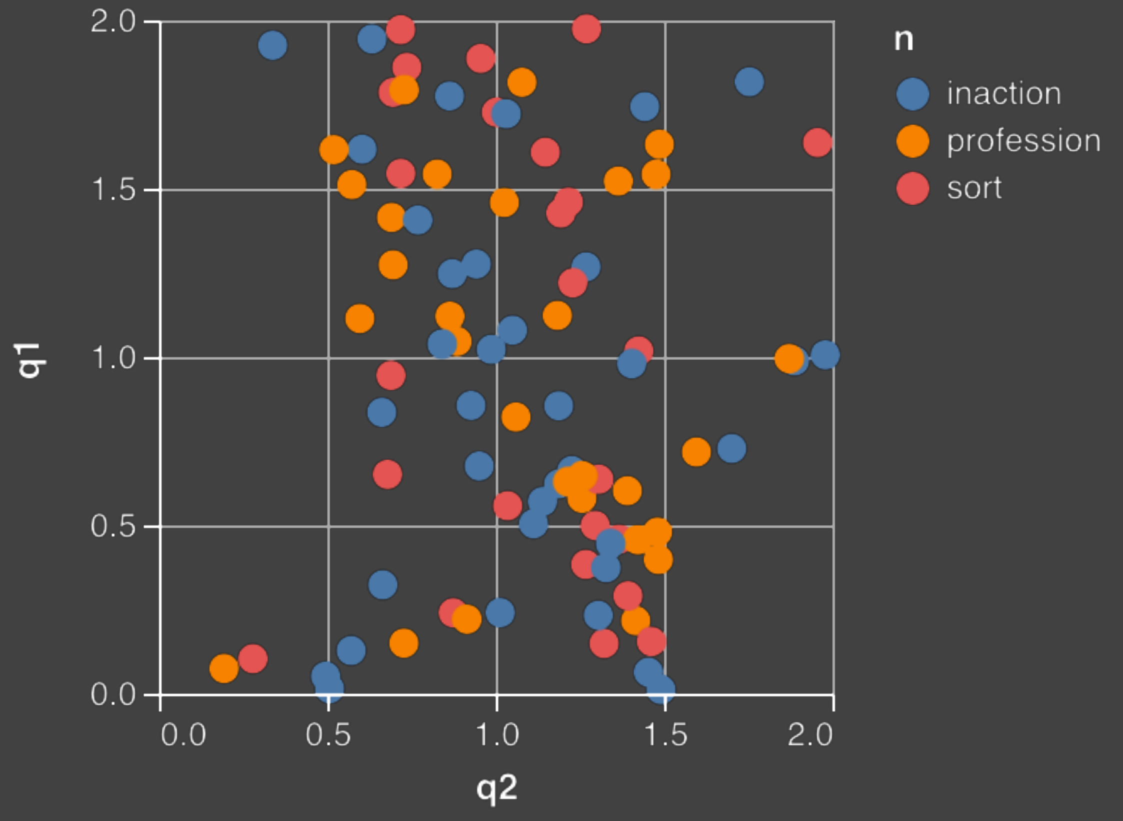
(no task awareness)

Task

Value



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

Draco-CQL

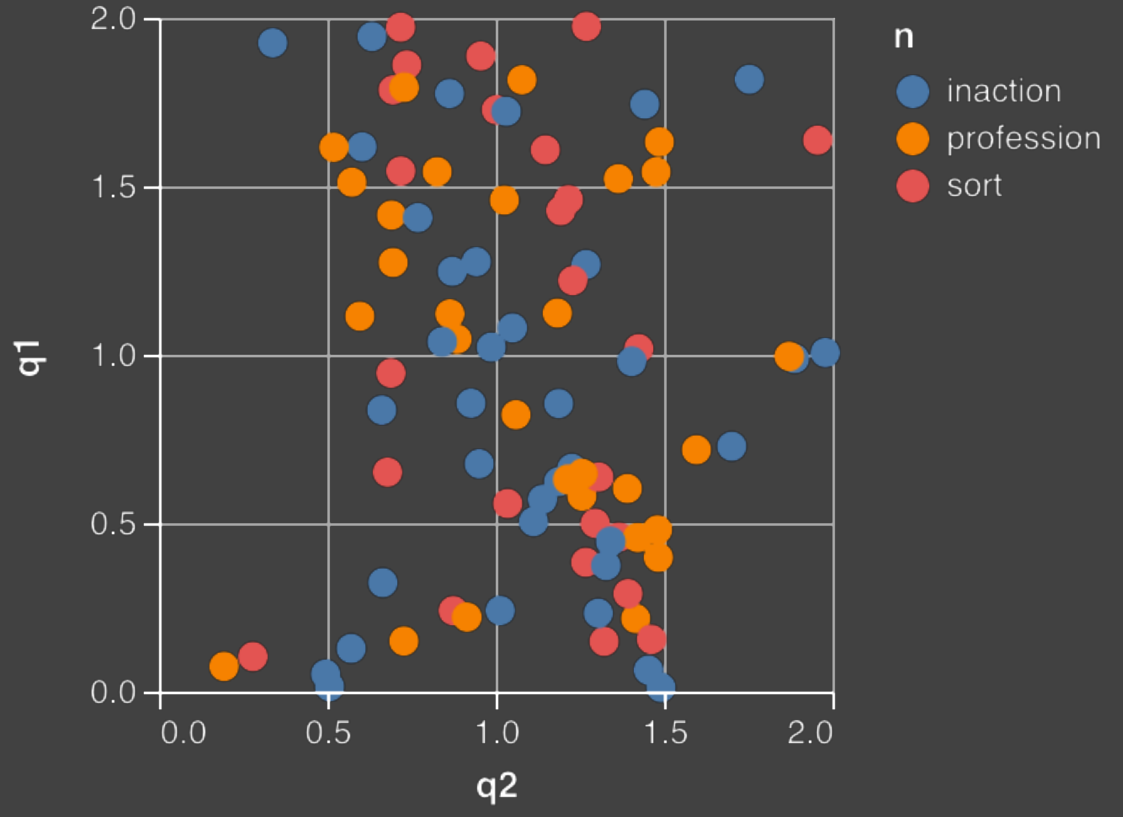
(no task awareness)

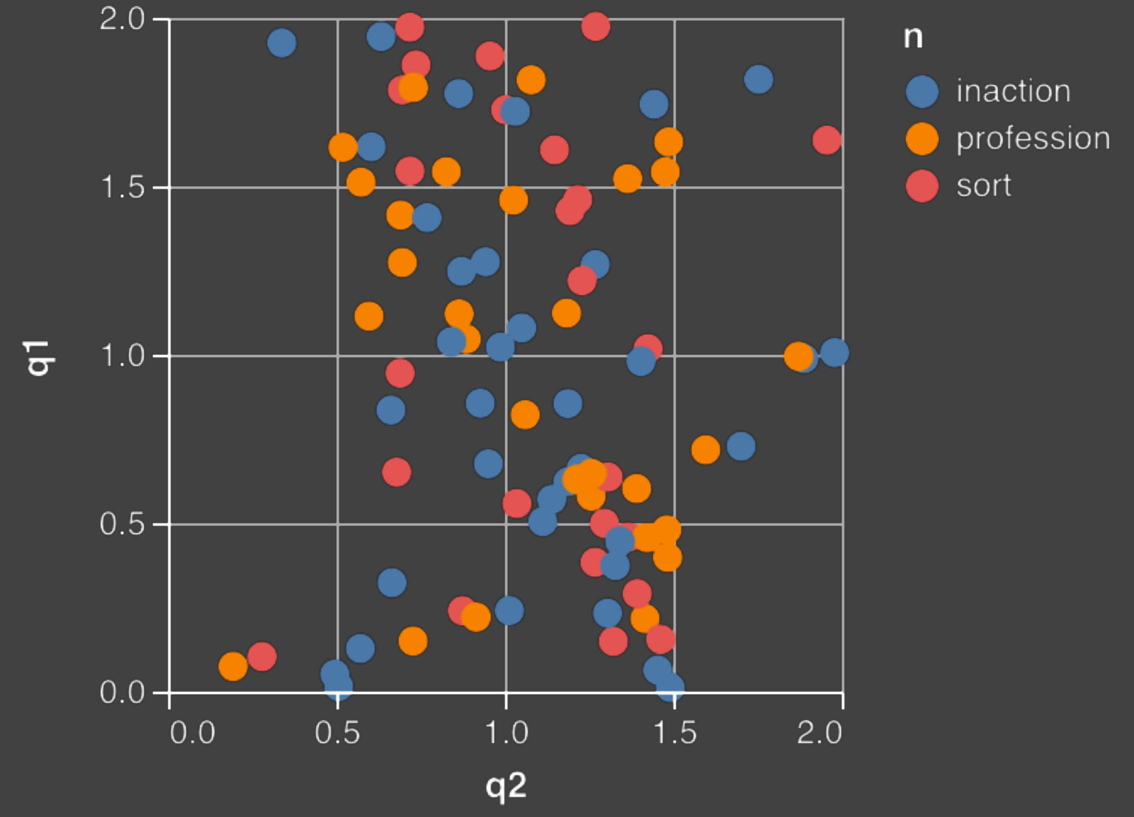
Draco-Learn

(task aware)

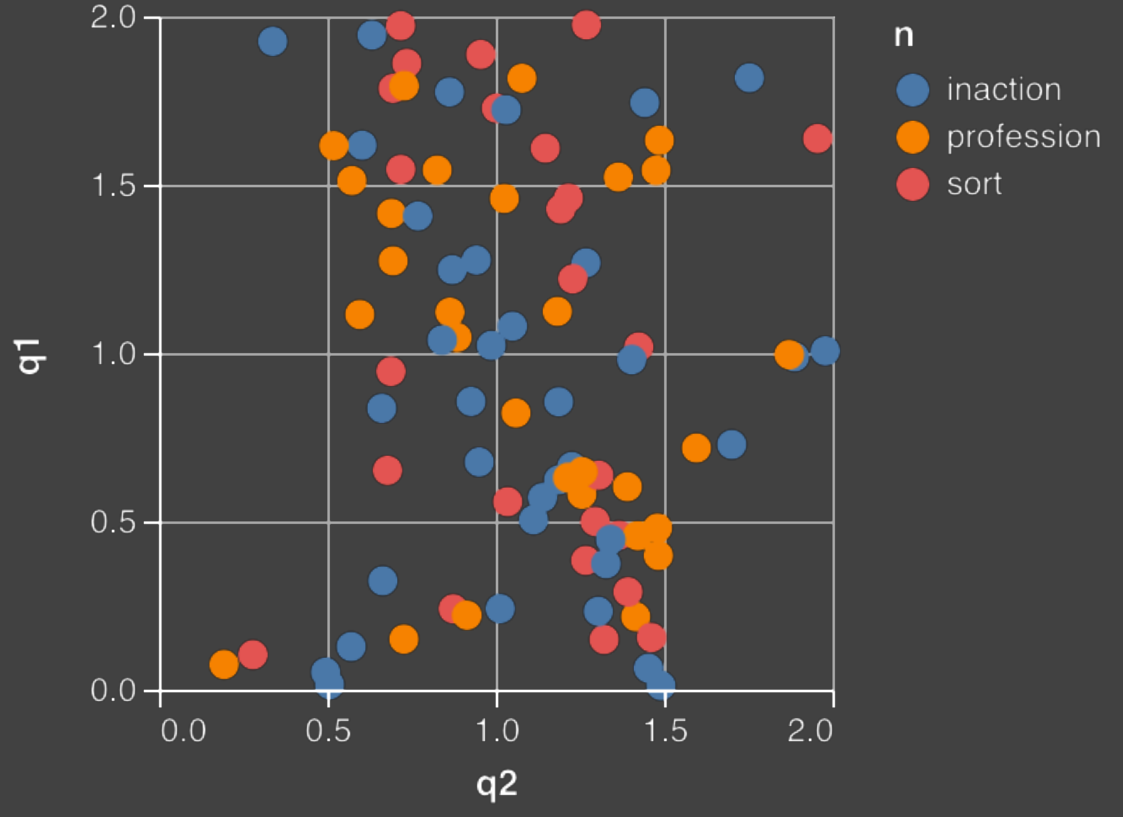
Task

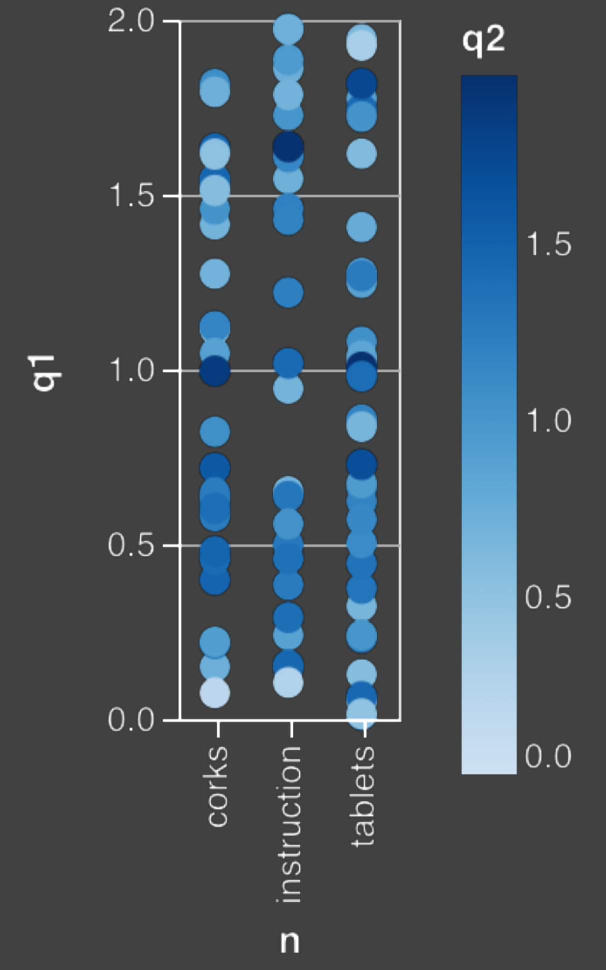
Value





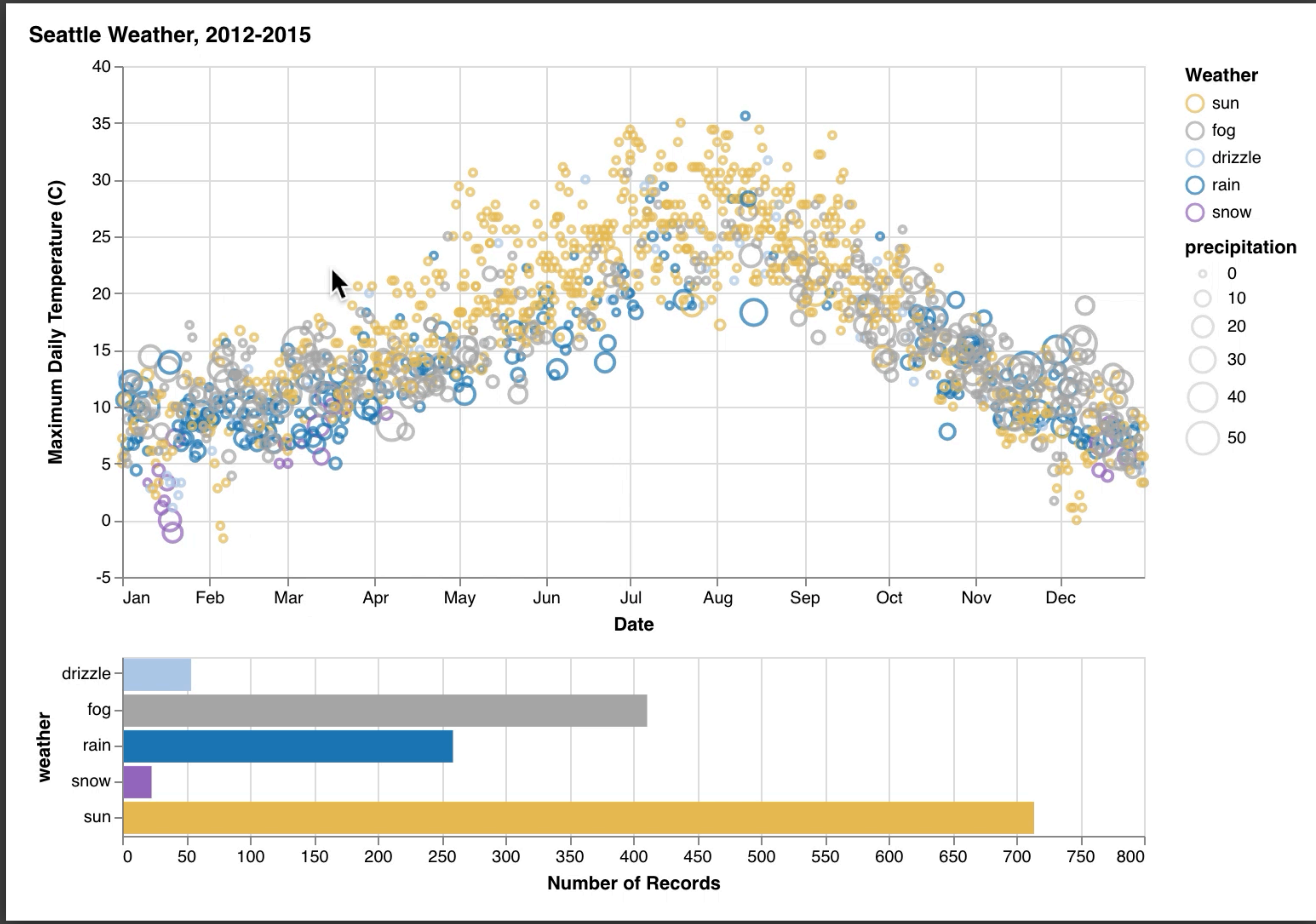
Summary





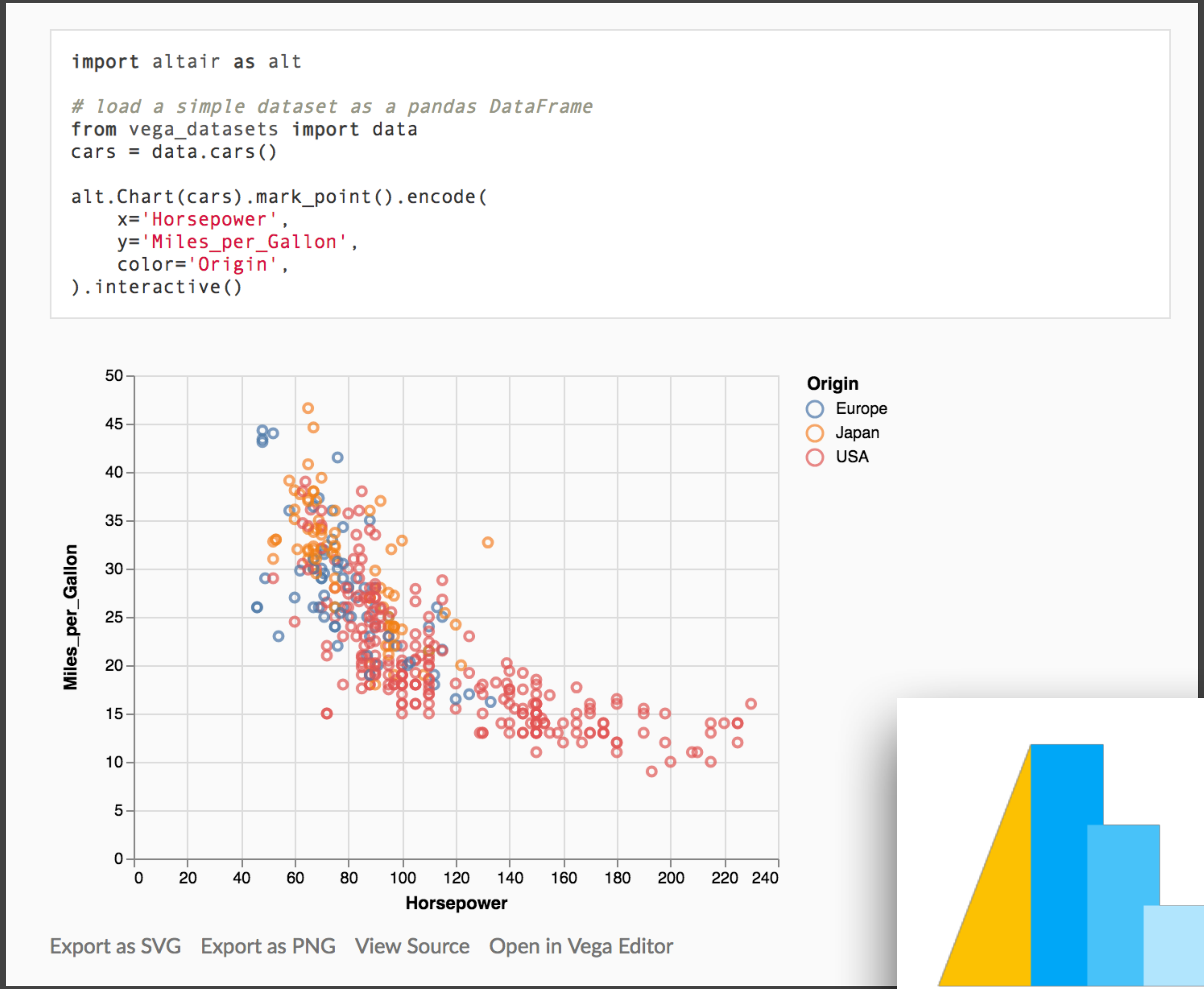
Future Directions

Future Directions



Automated Design for Interactive Dashboards

Future Directions

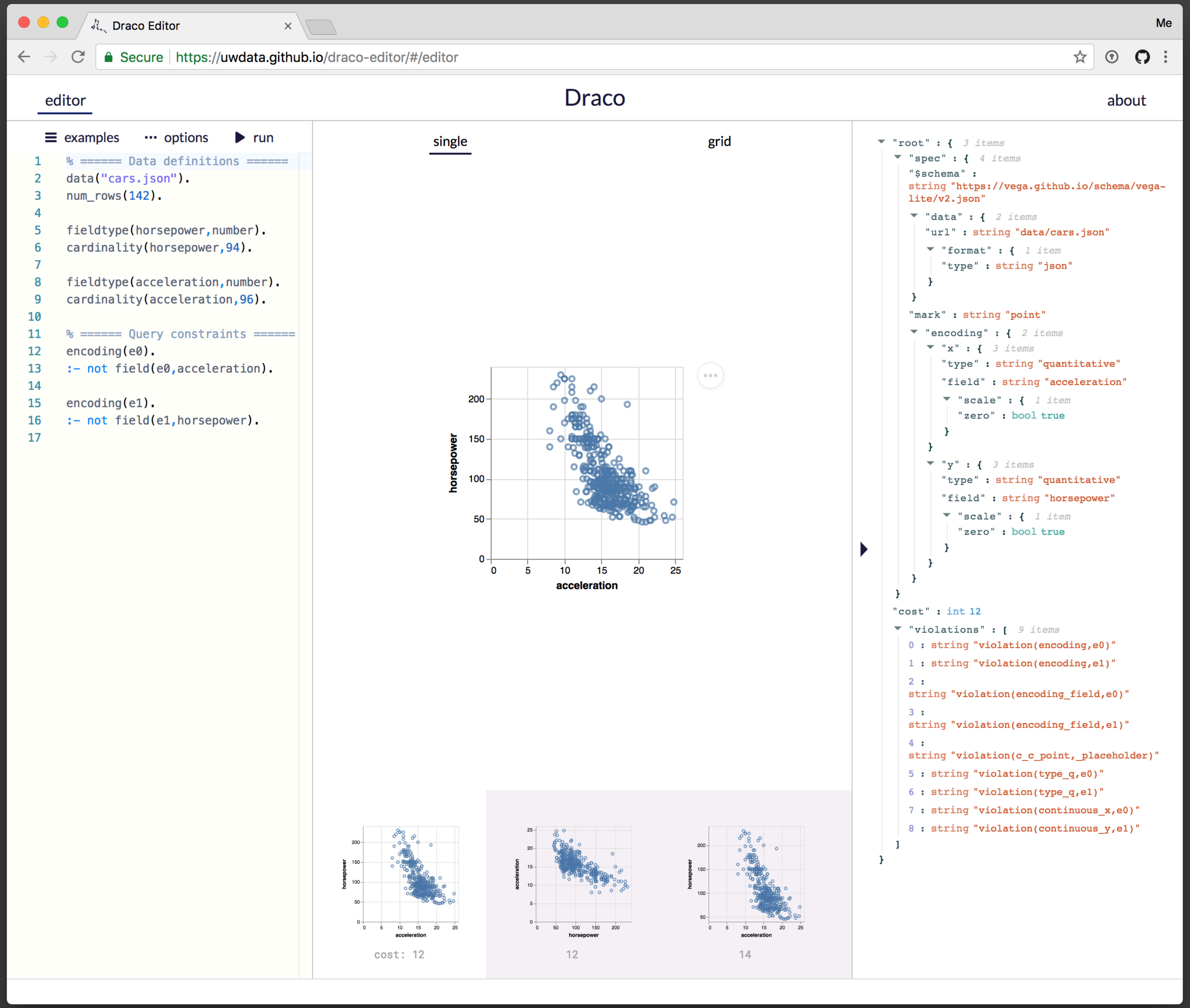


https://altair-viz.github.io/

Automated Design for Interactive Dashboards

Integration in Plotting APIs

Future Directions



uwdata.github.io/draco-editor

Automated Design for Interactive Dashboards

Integration in Plotting APIs

Tools to Browse, Update, and Compare Draco Knowledge Bases

Evaluate impact of new perceptual models

Draco Meetup

Thursday 13:00 at Shaan

meet at 12:50 at the revolving doors

Draco

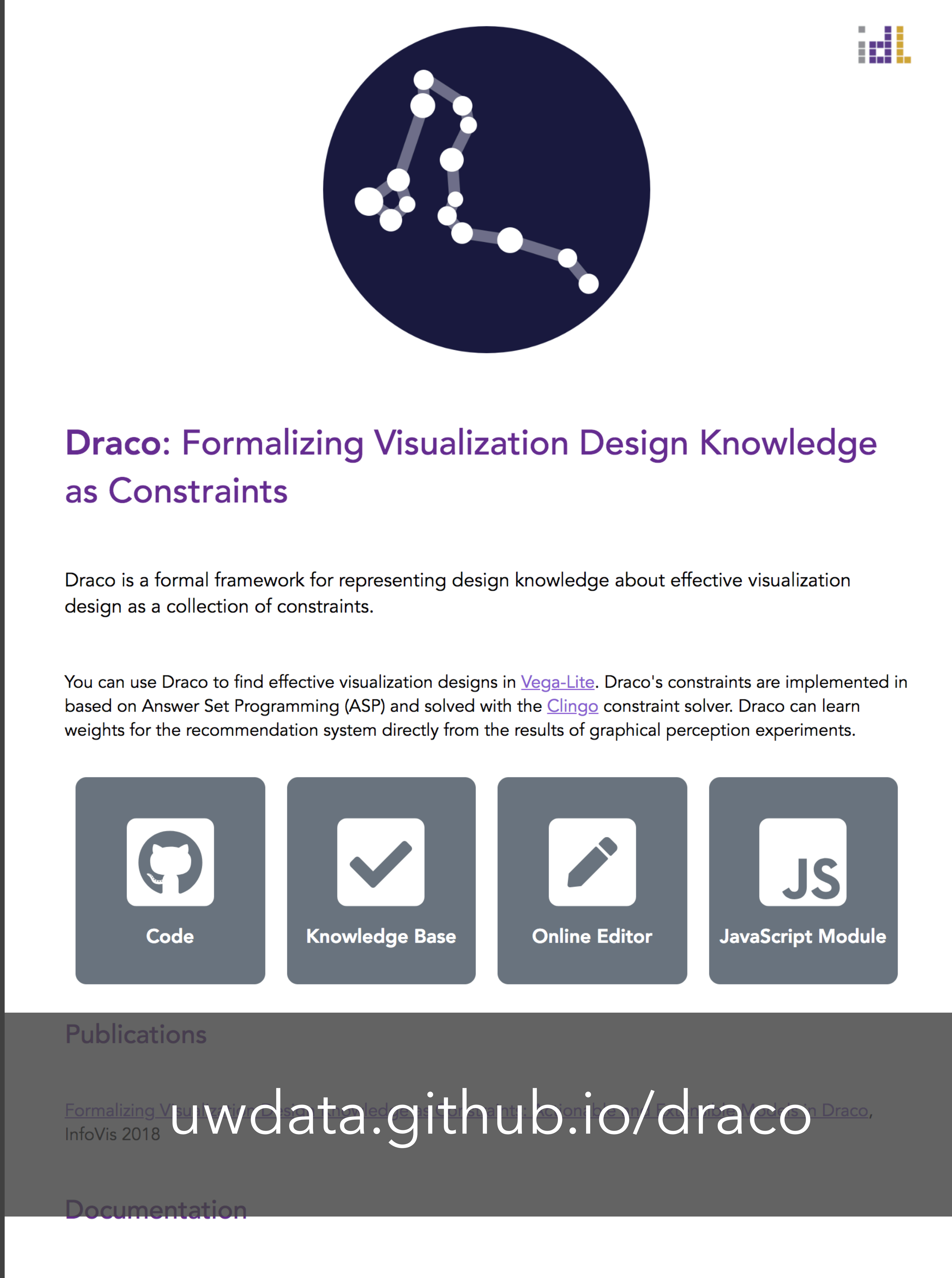
🧠 Extensible and Adaptive Knowledge Base of Visualization Design

📊 Automated Visualization Design Tool
Formal Reasoning
Shared Resource for Vis Community
Accelerate Knowledge Transfer

🤖 Learn Visualization Design

Dominik Moritz @domoritz et al.

@uwdata



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.

Code | Knowledge Base | Online Editor | JavaScript Module

Publications

Formalizing V... Draco, InfoVis 2018

Documentation

uwdata.github.io/draco