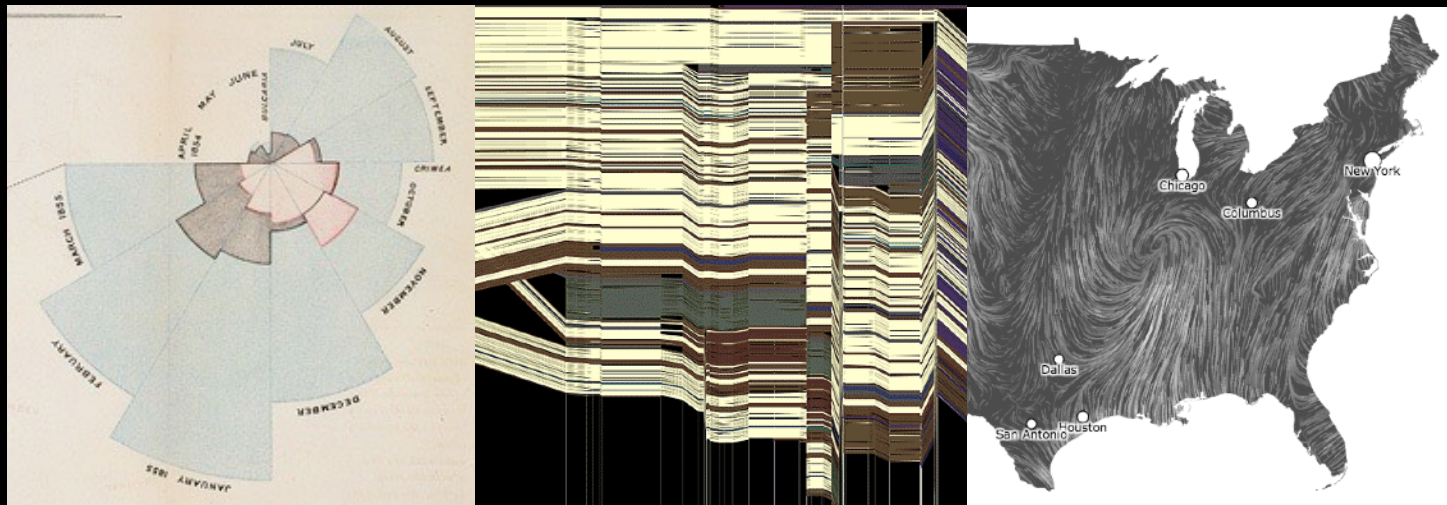


CSE 512 - Data Visualization

Visualization Tools, Part 2



Jeffrey Heer University of Washington

Visualization Tools

Ease-of-Use

Chart Typologies

Excel, Google Charts

Visual Analysis Grammars

ggplot2, Observable Plot, Vega-Lite

Visualization Libraries

Matplotlib, D3, Vega

Component Architectures

VTK, Prefuse

Graphics & Event APIs

Processing, OpenGL, Java2D

Expressiveness

Ease-of-Use

Chart Typologies

Excel, Google Charts

Visual Analysis Grammars

ggplot2, Observable Plot, Vega-Lite

Visualization Libraries

Matplotlib, D3, Vega

Component Architectures

VTK, Prefuse

Graphics & Event APIs

Processing, OpenGL, Java2D

Expressiveness

Grammar Building Blocks

Data

Input data to visualize

Transforms

Filter, aggregate, stats, layout

Scales

Map data values to visual values

Guides

Axes & legends to visualize scales

Marks

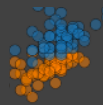
Data-representative graphics



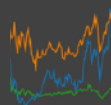
Area



Rect



Symbol



Line



Arc

Text

Chart Typologies

Excel, Google Charts

Visual Analysis Grammars

ggplot2, Observable Plot, Vega-Lite

Visualization Libraries

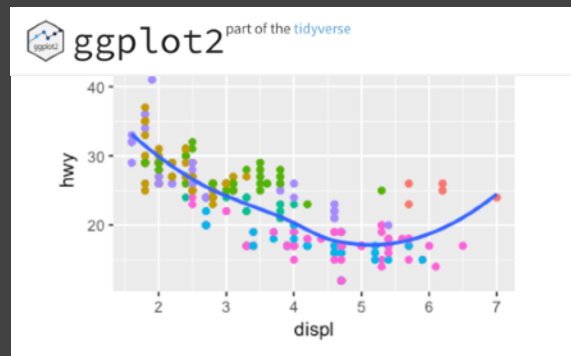
Matplotlib, D3, Vega

Component Architectures

VTK, Prefuse

Graphics & Event APIs

Processing, OpenGL, Java2D



Facilitate **rapid exploration** with **concise** specifications by omitting low-level details.

Infer **sensible defaults** and customize by overriding defaults.

Chart Typologies

Excel, Google Charts

Visual Analysis Grammars

ggplot2, Observable Plot, Vega-Lite

Visualization Libraries

Matplotlib, D3, Vega

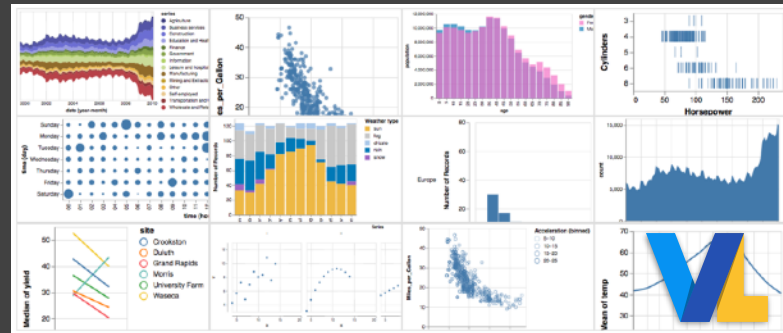
Component Architectures

VTK, Prefuse

Graphics & Event APIs

Processing, OpenGL, Java2D

Expressiveness



Facilitate **rapid exploration** with **concise** specifications by omitting low-level details.

Infer **sensible defaults** and customize by overriding defaults.

Chart Typologies

Excel, Google Charts

Visual Analysis Grammars

ggplot2, Observable Plot, Vega-Lite

Visualization Libraries

Matplotlib, D3, Vega

Component Architectures

VTK, Prefuse

Graphics & Event APIs

Processing, OpenGL, Java2D



Offer **fine-grained control** for composing interactive graphics.

But require **verbose** specifications and technical expertise.

d3.js Data-Driven Documents



Mike Bostock, Vadim Ogievetsky, Jeffrey Heer [TVCG 2011]
Jason Davies (d3-geo, 2011–13), Philippe Rivière (2016–)

What is D3?

1. A collection of reusable visualization utilities
2. A tool for updating the browser's Document Object Model (DOM) in response to input data

What is D3?

1. A collection of reusable visualization utilities

Data: d3.csv, d3.json, ...

Scales: d3.scaleLinear, d3.scaleLog, ...

Projections: d3.geoPath, d3.geoMercator, ...

Layout: d3.tree, d3.treemap, d3.force, ...

Interaction: d3.brush, d3.zoom, ...

2. A tool for updating the browser's Document Object Model (DOM) in response to input data

What is D3?

1. A collection of reusable visualization utilities
2. A tool for updating the browser's Document Object Model (DOM) in response to input data

Select: query DOM content

Join: bind input data to DOM elements

Update: set DOM element properties

Transition: animate changes over time

Why D3?

Enable highly custom visualization design

Support animation and dynamic displays

Support rich and varied interactions

Interoperate via web standards (HTML, SVG, CSS)

Avoid artificial limits. If a browser can do it, D3 should be able to take advantage of it.

Why D3?

"the authors have undeniably helped to bring data visualization to the mainstream. [D3] is a cornerstone contribution to this conference specifically and more generally to the success of our field as a whole"

IEEE VIS 2021 Test of Time Award

Why D3?

D3 “slingshotted the field into growth, diversification and creativity that has been unprecedented” and “changed how millions of data visualizations are created across newsrooms, websites, and personal portfolios”

Information is Beautiful 2022 Test of Time Award

Why D3?

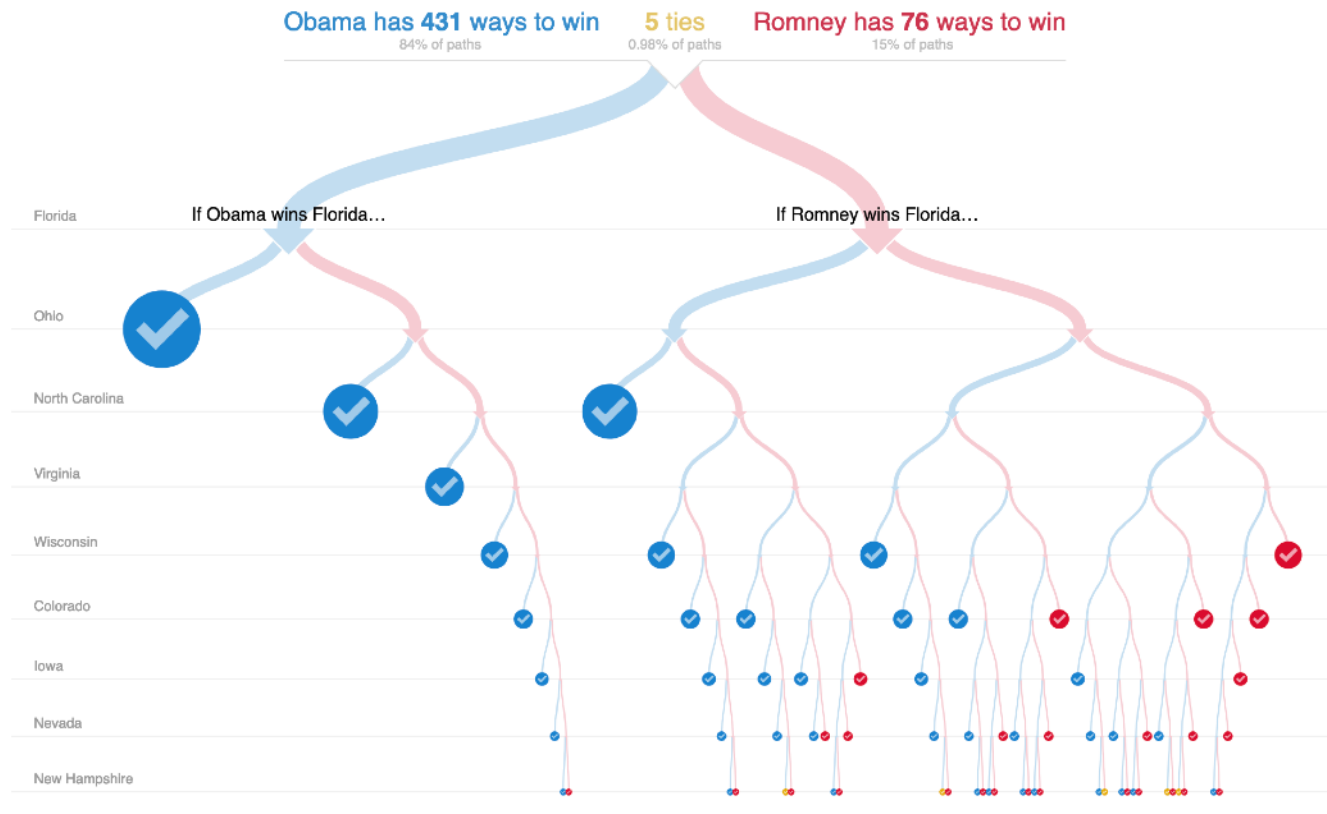
“Use D3 if you think it’s perfectly normal to write a hundred lines of code for a bar chart.”

Amanda Cox, Former Graphics Editor, NY Times

512 Paths to the White House

Select a winner in the most competitive states below to see all the paths to victory available for either candidate.

Fla.	Ohio	N.C.	Va.	Wis.	Colo.	Iowa	Nev.	N.H.
Dem Rep	Dem Rep	Dem Rep	Dem Rep	Dem Rep	Dem Rep	Dem Rep	Dem Rep	Dem Rep



D3 Selections

The core abstraction in D3 is a *selection*.

D3 Selections

The core abstraction in D3 is a ***selection***.

```
// Add and configure an SVG element (<svg width="500" height="300">)  
svg = d3.append("svg")           // add new SVG to page body  
    .attr("width", 500)          // set SVG width to 500px  
    .attr("height", 300);        // set SVG height to 300px
```

Data

DOM

Data

```
svg = d3.append("svg")  
  .attr("width", 500)  
  .attr("height", 300);
```

DOM

<svg width="500" ...>

</svg>

D3 Selections

The core abstraction in D3 is a ***selection***.

```
// Add and configure an SVG element (<svg width="500" height="300">)
```

```
svg = d3.append("svg")           // add new SVG to page body
    .attr("width", 500)          // set SVG width to 500px
    .attr("height", 300);        // set SVG height to 300px
```

```
// Select & update existing rectangles contained in the SVG element
```

```
svg.selectAll("rect")            // select all SVG rectangles
    .attr("width", 100)          // set rect widths to 100px
    .style("fill", "steelblue"); // set rect fill colors
```

Data

DOM

<svg width="500" ...>

</svg>

Data

```
svg.selectAll("rect")
```

DOM

```
<svg width="500" ...>
```

???

```
</svg>
```

Data

DOM

```
<svg width="500" ...>  
  <rect ..></rect>  
  <rect ..></rect>  
  <rect ..></rect>  
  <rect ..></rect>  
  <rect ..></rect>  
</svg>
```

Data

```
svg.selectAll("rect")
```

DOM

```
<svg width="500" ...>
```

```
<rect ... />
```

```
<rect ... />
```

```
<rect ... />
```

```
<rect ... />
```

```
<rect ... />
```

```
</svg>
```

Data

```
svg.selectAll("rect")  
  .attr("width", 100)  
  .style("fill", "steelblue")
```

DOM

```
<svg width="500" ...>  
  <rect width="100"  
    style="fill: steelblue;"  
  />  
  <rect width="100"  
    style="fill: steelblue;"  
  />  
  <rect width="100"  
    style="fill: steelblue;"
```

Data Binding

Selections can *bind* data and **DOM** elements.

```
values = [ {...}, {...}, {...}, ... ]; // input data as JS objects
```

Data Binding

Selections can *bind* data and **DOM** elements.

```
values = [ {...}, {...}, {...}, ... ]; // input data as JS objects
```

```
// Select SVG rectangles and bind them to data values.
```

```
bars = svg.selectAll("rect.bars").data(values);
```

Data

```
values = [
  { cat: "a", value: 5 },
  { cat: "b", value: 7 },
  { cat: "c", value: 3 },
  { cat: "d", value: 4 },
  { cat: "e", value: 6 }
];
```

DOM

```
<svg width=500 ...>
```

</svg>

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>
```

?? ??

```
</svg>
```

```
bars = svg.selectAll("rect").data(values)
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

?
?
?
?
?

DOM

```
<svg width=500 ...>
```

```
</svg>
```

```
bars = svg.selectAll("rect").data(values)
```

Data Binding

Selections can *bind* data and **DOM** elements.

```
values = [ {...}, {...}, {...}, ... ]; // input data as JS objects
```

```
// Select SVG rectangles and bind them to data values.
```

```
bars = svg.selectAll("rect.bars").data(values);
```

```
// What if the DOM elements don't exist yet? The enter set represents data
```

```
// values that do not yet have matching DOM elements.
```

```
bars.enter().append("rect").attr("class", "bars");
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

?
?
?
?
?

DOM

```
<svg width=500 ...>
```

```
</svg>
```

```
bars = svg.selectAll("rect").data(values)
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect />  
  <rect />  
  <rect />  
  <rect />  
  <rect />  
</svg>
```

```
bars.enter().append("rect")
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
</svg>
```

```
bars.enter().append("rect").attr("class", "bars")
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect x="..." />  
  <rect x="..." />  
  <rect x="..." />  
  <rect x="..." />  
  <rect x="..." />  
</svg>
```

```
bars.enter().append("rect")  
  .attr("x", d => xscale(d.cat))
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect height="..." />  
  <rect height="..." />  
  <rect height="..." />  
  <rect height="..." />  
  <rect height="..." />  
</svg>
```

```
bars.enter().append("rect")  
  .attr("height", d => yscale(d.value))
```

Data Binding

Selections can *bind* data and **DOM** elements.

```
values = [ {...}, {...}, {...}, ... ]; // input data as JS objects
```

```
// Select SVG rectangles and bind them to data values.
```

```
bars = svg.selectAll("rect.bars").data(values);
```

```
// What if the DOM elements don't exist yet? The enter set represents data
```

```
// values that do not yet have matching DOM elements.
```

```
bars.enter().append("rect").attr("class", "bars");
```

```
// What if data values are removed? The exit set is a selection of existing
```

```
// DOM elements who no longer have matching data values.
```

```
bars.exit().remove();
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
</svg>
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "b", value: 7 },  
  { cat: "c", value: 3 },  
  { cat: "d", value: 4 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
</svg>
```

```
values.filter(d => ![ 'b', 'd' ].includes(d.cat))
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "c", value: 3 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
</svg>
```

```
bars = svg.selectAll("rect.bars").data(values)
```

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "c", value: 3 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
</svg>
```

bars.exit()

Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "c", value: 3 },  
  { cat: "e", value: 6 }  
];
```

DOM

```
<svg width=500 ...>  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
</svg>
```

bars.exit().remove()

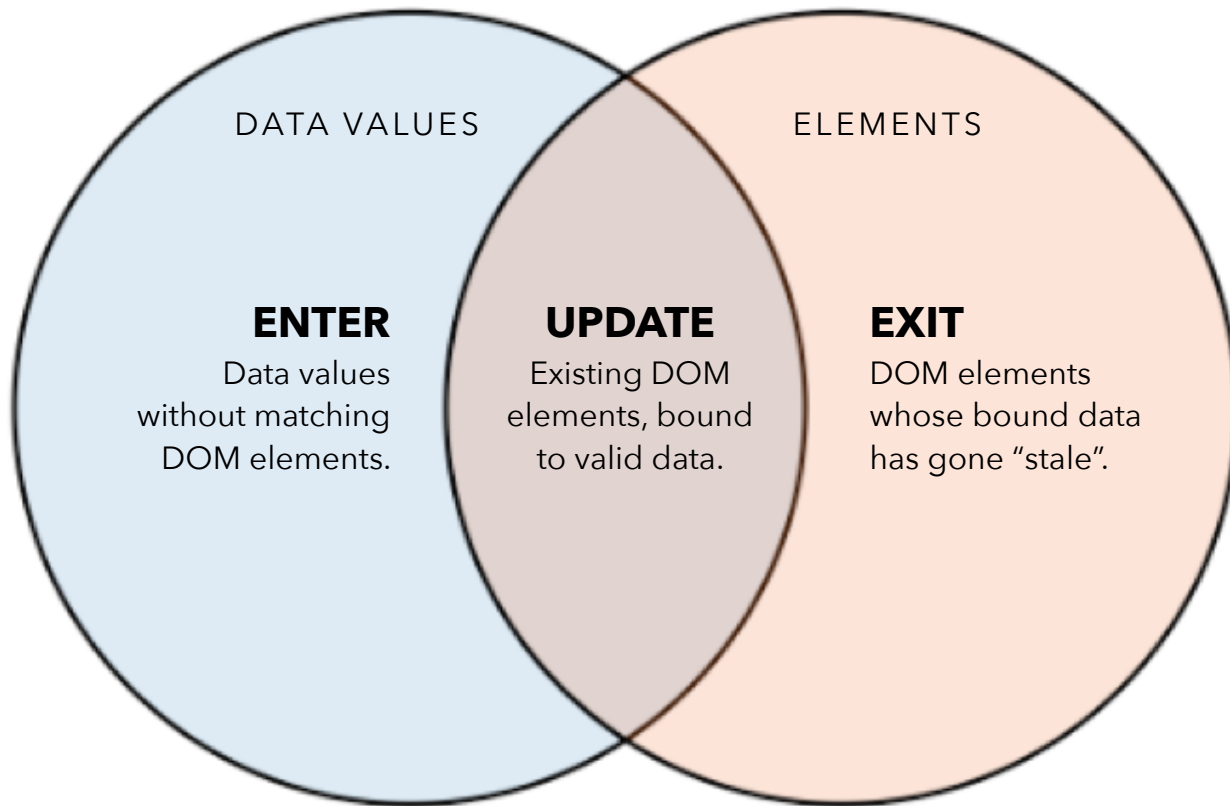
Data

```
values = [  
  { cat: "a", value: 5 },  
  { cat: "c", value: 3 },  
  { cat: "e", value: 6 }  
];
```

DOM

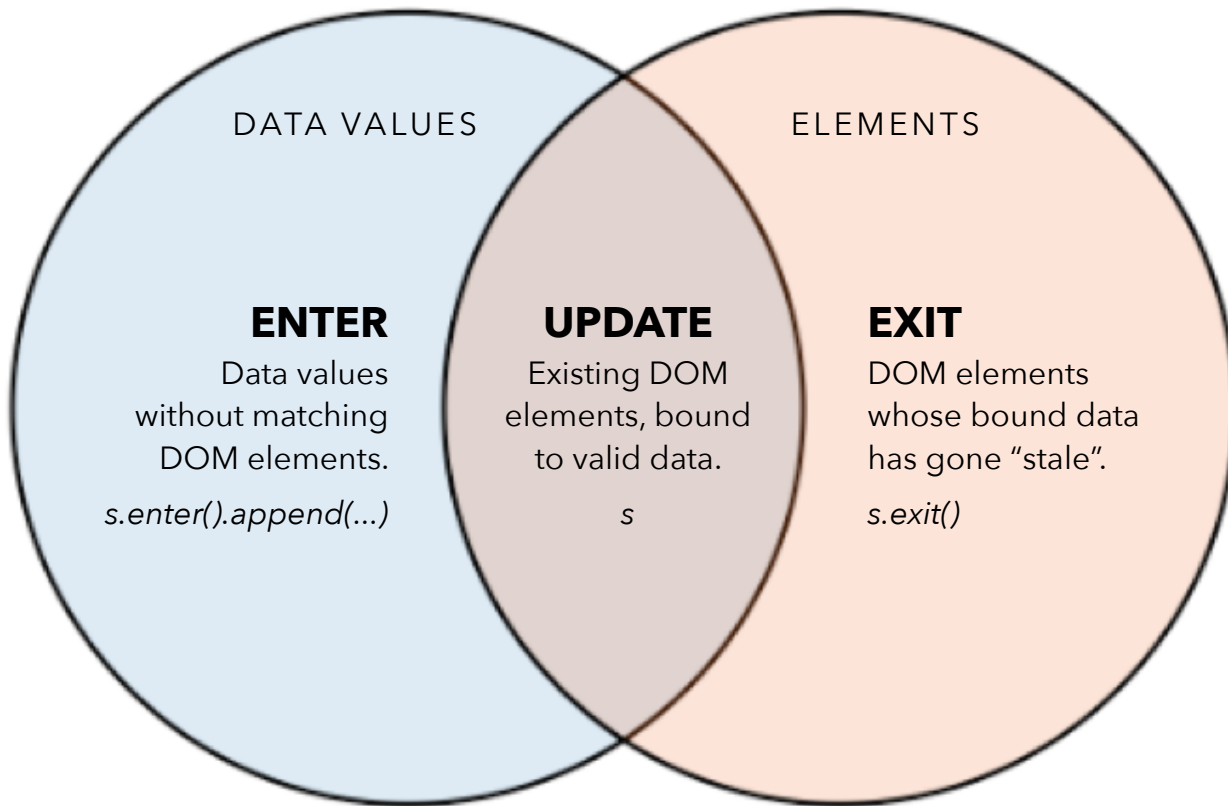
```
<svg width=500 ...>  
  <rect class="bars" />  
  <rect class="bars" />  
  <rect class="bars" />  
</svg>
```

The Data Join



The Data Join

```
var s = d3.selectAll(...).data(...)
```



Data Binding

Selections can *bind* data and **DOM** elements.

```
values = [ {...}, {...}, {...}, ... ]; // input data as JS objects
```

```
// Select SVG rectangles and bind them to data values.
```

```
bars = svg.selectAll("rect.bars").data(values)
    .join(
        enter => enter.append("rect"), // create new
        update => update,                // update current
        exit => exit.remove()            // remove outdated
    )
```

D3 Modules

Data Parsing / Formatting (JSON, CSV, ...)

Shape Helpers (arcs, curves, areas, symbols, ...)

Scale Transforms (linear, log, ordinal, ...)

Color Spaces (RGB, HSL, LAB, ...)

Animated Transitions (tweening, easing, ...)

Geographic Mapping (projections, clipping, ...)

Layout Algorithms (stack, pie, force, trees, ...)

Interactive Behaviors (brush, zoom, drag, ...)

Many of these correspond to future lecture topics!

Administrivia

A2: Deceptive Visualization

Design **two** static visualizations for a dataset.

1. An *earnest* visualization that accurately conveys the data.
2. A *deceptive* visualization that misrepresents the data.

Your two visualizations should address different questions.

Then, design a deceptive visualization that appears to be honest. What does it tell you about your classmates and course staff?

You are free to create your own dataset, but we have also provided some preselected datasets for you.

Submit your two images and a brief write-up on Gradescope.

Due by **Mon 4/21 EOD**.

A2: Peer Reviews

You will be assigned two peer W2 submissions to review.
For each:

- Try to determine which is earnest and which is deceptive
- Share a rationale for how you made this determination
- Share feedback using the “I Like / I Wish / What If” rubric

Assigned reviews will be posted to a A2 Peer Review thread on Ed, along with a link to a Google Form. You should submit two forms: one for each A2 peer review.

Due by **Tue 4/29 EOD.**

I Like... / I Wish... / What If?

I LIKE...

Praise for design ideas and/or well-executed implementation details.

Example: "I like the navigation through time via the slider; the patterns observed as one moves forward are compelling!"

I WISH...

Constructive statements on how the design might be improved or further refined. *Example: "I wish moving the slider caused the visualization to update immediately, rather than the current lag."*

WHAT IF?

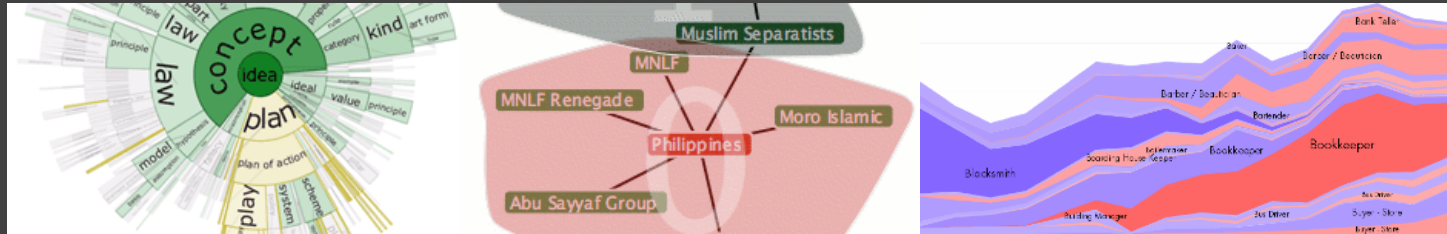
Suggest alternative design directions, or even wacky half-baked ideas.

Example: "What if we got rid of the slider and enabled direct manipulation navigation by dragging data points directly?"

A3: Interactive Visualization

Create an interactive visualization in a team of 1-3 people. Choose a dataset and a driving question, develop a visualization + interaction techniques, then deploy your visualization on the web.

1. Form team, topic & data and start prototyping.
2. Complete implementation and submit to Gradescope by *EOD* on **Monday, May 12**.

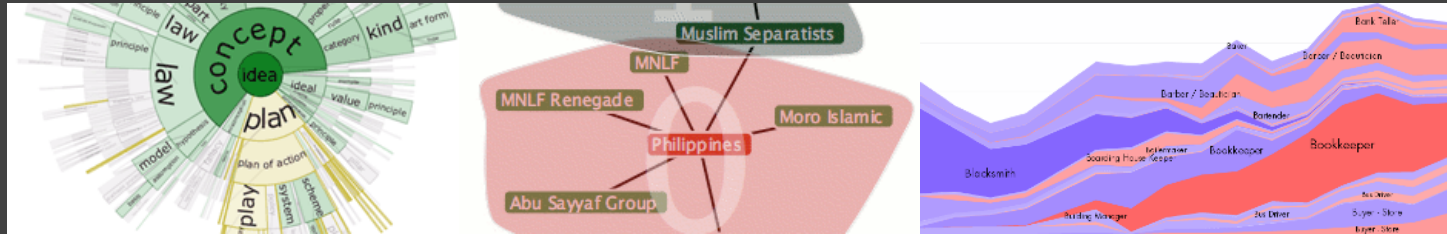


Form a Project Team

Form a **team of 1-4 people** for the A3 assignment.
Submit team composition by **Friday May 2**.

If you're looking for team mates, you can post on Ed about your interests/skills/project ideas!

You may continue with the same team for the final project, or form a new team later. It's up to you.

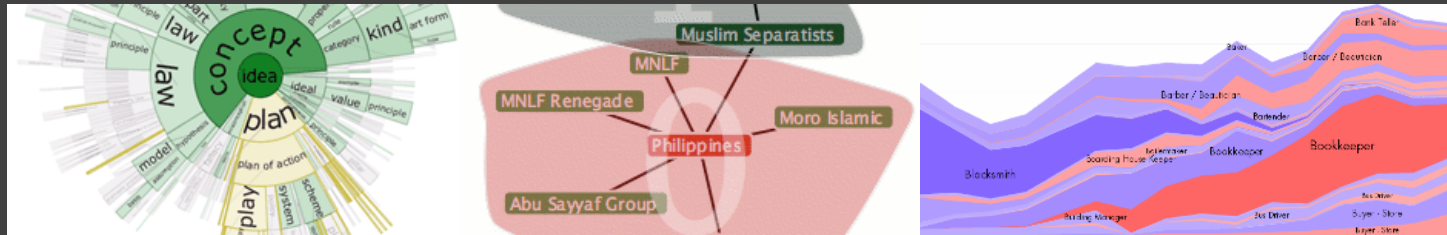


Requirements

Interactive. You must implement interaction methods!
However, this is not only selection / filtering / tooltips.
Also consider annotations or other narrative features to
draw attention and provide additional context

Web-based. D3/Vega-Lite are encouraged, but not required. Deploy to web using GitLab pages.

Write-up. Provide design rationale.

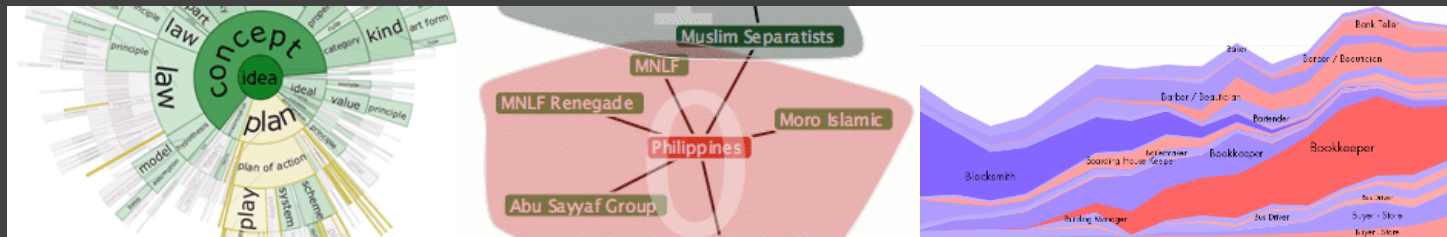


Interactive Prototype Tips

Start now. It will take longer than you think.

Keep it simple. Choose a *minimal* set of interactions that enables users to explore and generate interesting insights. Do not feel obligated to convey *everything* about the data: focus on a compelling subset.

Promote engagement. How do your chosen interactions reveal interesting observations?



A Visualization Tool Stack

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2, Vega-Lite

Visualization Grammars

D3.js, Vega

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Canvas, OpenGL, Processing

Chart Typologies

Excel, Many Eyes, Google Charts

Charting
Tools

Visual Analysis Grammars

VizQL, ggplot2, Vega-Lite

Declarative
Languages

Visualization Grammars

D3.js, Vega

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Canvas, OpenGL, Processing

Chart Typologies

Excel, Many Eyes, Google Charts

Charting
Tools

Visual Analysis Grammars

VizQL, ggplot2, Vega-Lite

Declarative
Languages

Visualization Grammars

D3.js, Vega

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Canvas, OpenGL, Processing

What is a Declarative Language?

Programming by describing *what*, not *how*

Separate **specification** (*what you want*) from **execution** (*how it should be computed*)

In contrast to **imperative programming**, where you must give explicit steps.

What is a Declarative Language?

Programming by describing *what*, not *how*

Separate **specification** (*what you want*) from **execution** (*how it should be computed*)

In contrast to **imperative programming**, where you must give explicit steps.

```
d3.selectAll("rect")  
  .data(my_data)  
  .join("rect")  
  .attr("x", d => xscale(d.foo))  
  .attr("y", d => yscale(d.bar))
```

Switch to Global Edition,

JOBS
REAL ESTATE
AUTOS
ALL CLASSIFIEDS

WORLD
U.S.
POLITICS
N.Y./REGION
BUSINESS
TECHNOLOGY
SPORTS
SCIENCE
HEALTH
OPINION

ARTS
Books
Movies
Music
Television

— 2010 Midterm Elections —

Tea Party Vow to Deter Voter Fraud Is Called Scare Tactic

By IAN URBINA 2:19 PM ET
Voting rights group say that Tea Party members' plan to question voters' eligibility at the polls is intended to suppress minority and poor voters.

Post a Comment | Read (355)

Glaxo Pays \$750 Million Fine for Tainted Products

By GARDNER HARRIS and DUFF



Joshua Kirtzel for The New York Times

Painting at 99, With No Compromises

By ROBIN FINN
An exhibition celebrating Will Barnett's centennial year traces his evolution as a modern American artist.

OPINION »
OP-ED CONTRIBUTOR
Humans to Asteroids: Watch Out!
How to keep near-Earth objects from hitting us.



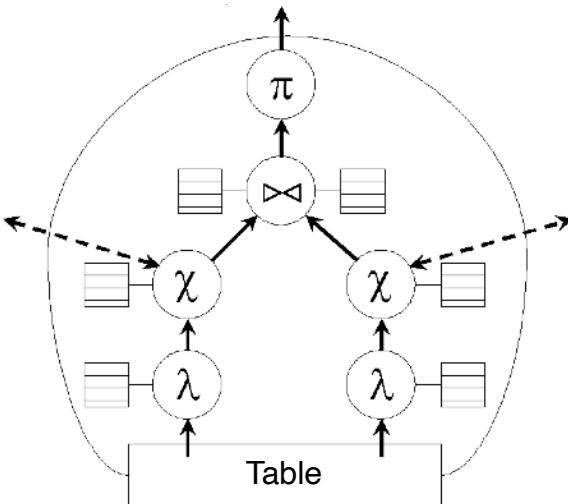
- Brooks: No Second Thoughts | Comments (200)
- Herbert: The Corrosion of America
- Cohen: Turkey Steps Out
- Editorial: Mortgage Mess
- Bloggingheads: Jon Stewart's Power

MARKETS » AI 3:56 PM ET
S.&P. 500 Dow Nasdaq

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!--[if IE]><![endif]-->
<html>
<head>...</head>
<body id="home" style="visibility: visible;">
<script src="http://connect.facebook.net/en_US/all.js"></script>
<div id="fb-root"></div>
<a name="top"></a>
<div id="shell">
<ul id="memberTools">...</ul>
<!-- ADXINFO classification="text_ad" campaign="nyt2010-circ -->
<div class="tabsContainer">...</div>
<!-- close .tabsContainer -->
<div id="page" class="tabContent active">...</div>
<!--close page -->
</div>
<!--close shell -->
<script type="text/javascript" language="JavaScript">...</script>
</span>
<script type="text/javascript">...</script>

<script type="text/javascript" src="http://graphics8.nytimes.com/ads/blank.gif">
</script>
```

HTML / CSS



```
SELECT customer_id, customer_name,
COUNT(order_id) as total
FROM customers
INNER JOIN orders ON
customers.customer_id
= orders.customer_id
GROUP BY customer_id, customer_name
HAVING COUNT(order_id) > 5
ORDER BY COUNT(order_id) DESC
```

SQL

Why Declarative Languages?

Faster iteration, less code, larger user base?

Better visualization. *Smart defaults.*

Reuse. *Write-once, then re-apply.*

Performance. *Optimization, scalability.*

Portability. *Multiple devices, renderers, inputs.*

Programmatic generation.

Write programs which output visualizations.

Automated search & recommendation.

Chart Typologies

Excel, Many Eyes, Google Charts

Charting
Tools

Visual Analysis Grammars

VizQL, ggplot2, *Vega-Lite*

Declarative
Languages

Visualization Grammars

D3.js, *Vega*

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Processing, OpenGL, Java2D

Chart Typologies

Excel, Many Eyes, Google Charts



Charting
Tools

Visual Analysis Grammars

VizQL, ggplot2, *Vega-Lite*

Visualization Grammars

D3.js, *Vega*

Declarative
Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Processing, OpenGL, Java2D

Visual Analysis Grammars

VizQL, ggplot2, *Vega-Lite*

Declarative
Languages

Visualization Grammars

D3.js, *Vega*

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Processing, OpenGL, Java2D

Interactive Data Exploration

Tableau, *Lyra*, *Voyager*

Graphical
Interfaces

Visual Analysis Grammars

VizQL, ggplot2, *Vega-Lite*

Declarative
Languages

Visualization Grammars

D3.js, *Vega*

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Processing, OpenGL, Java2D

The Lyra Visualization Design Environment (VDE) ^{alpha}

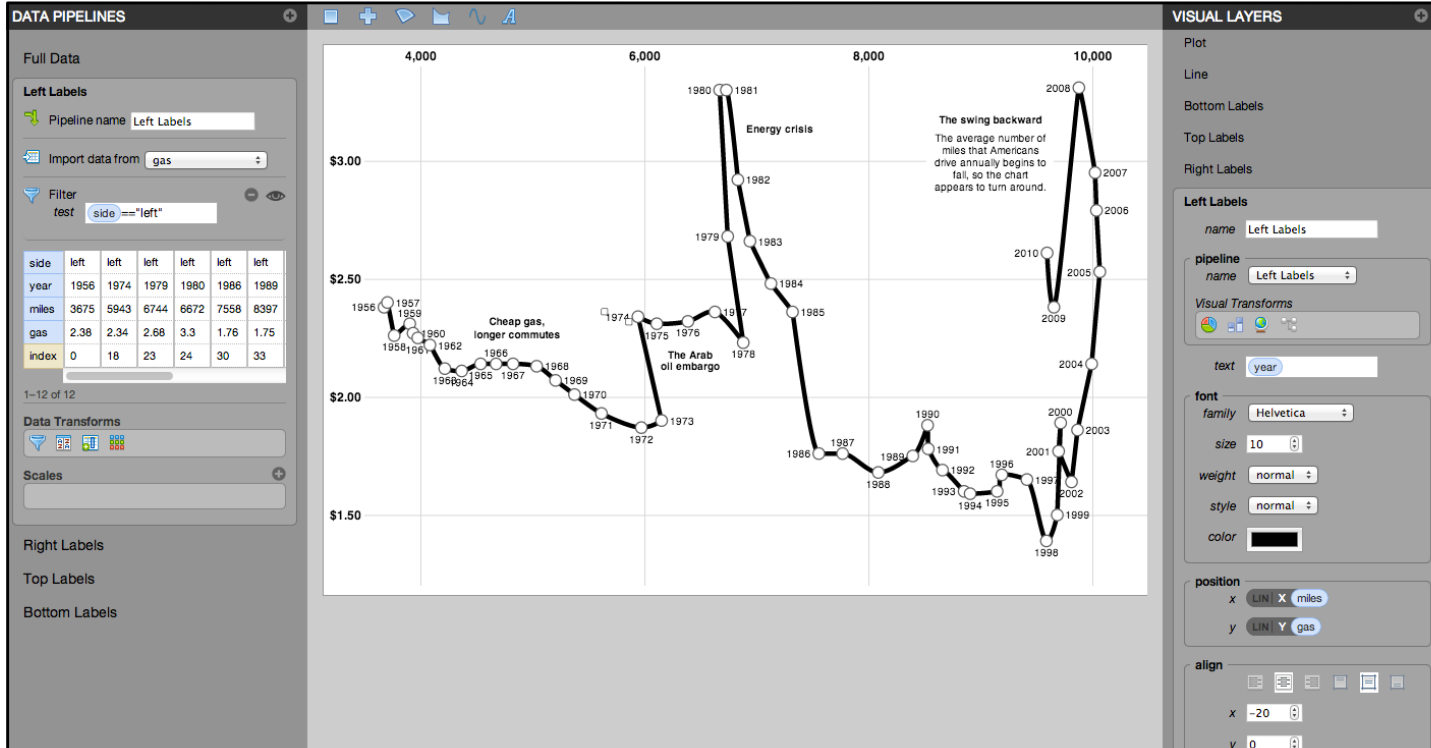
Arvind Satyanarayan, Kanit "Ham" Wongsuphasawat, Jeffrey Heer



William Playfair's classic chart comparing the price of wheat and wages in England recreated in the Lyra VDE.

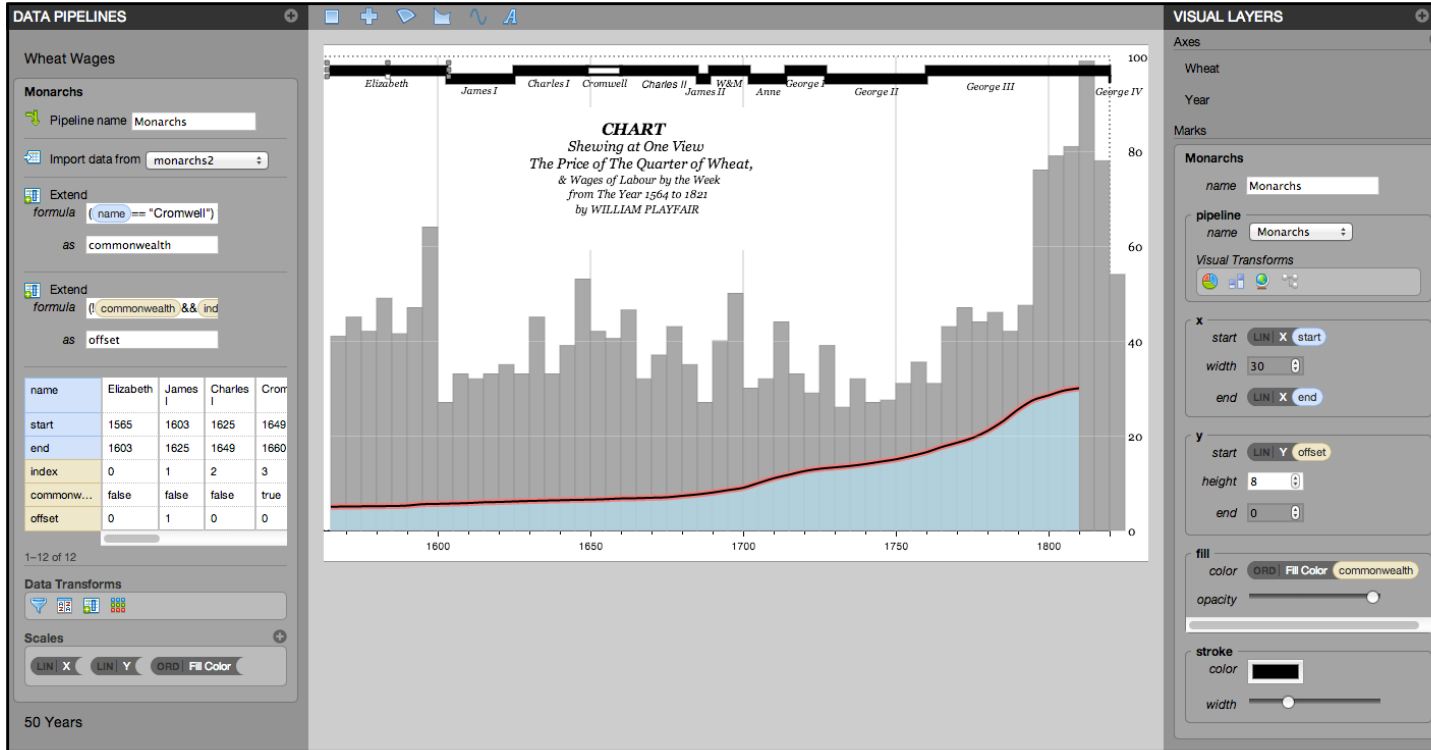
See also: Charticulator, Data Illustrator

Lyra A Visualization Design Environment



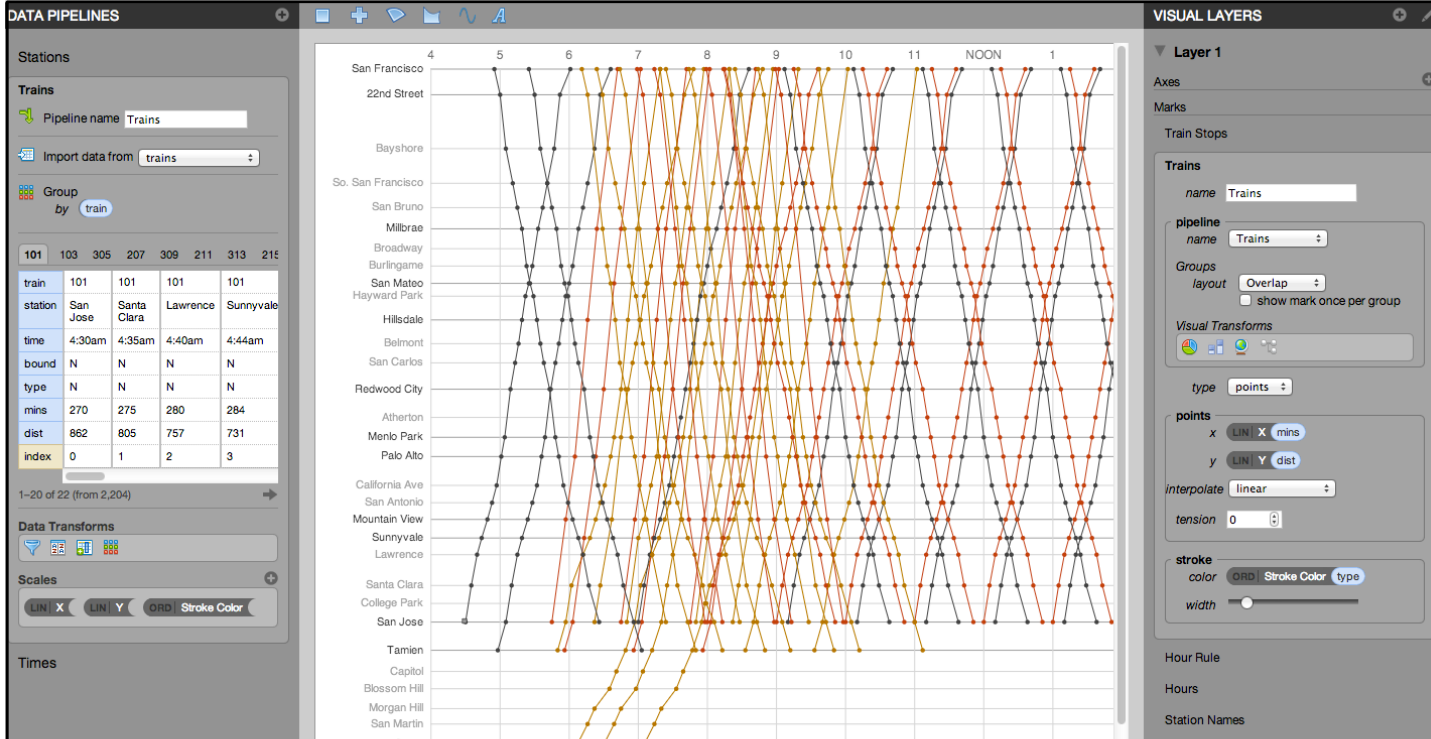
Driving Shifts into Reverse by Hannah Fairfield, NYTimes

Lyra A Visualization Design Environment



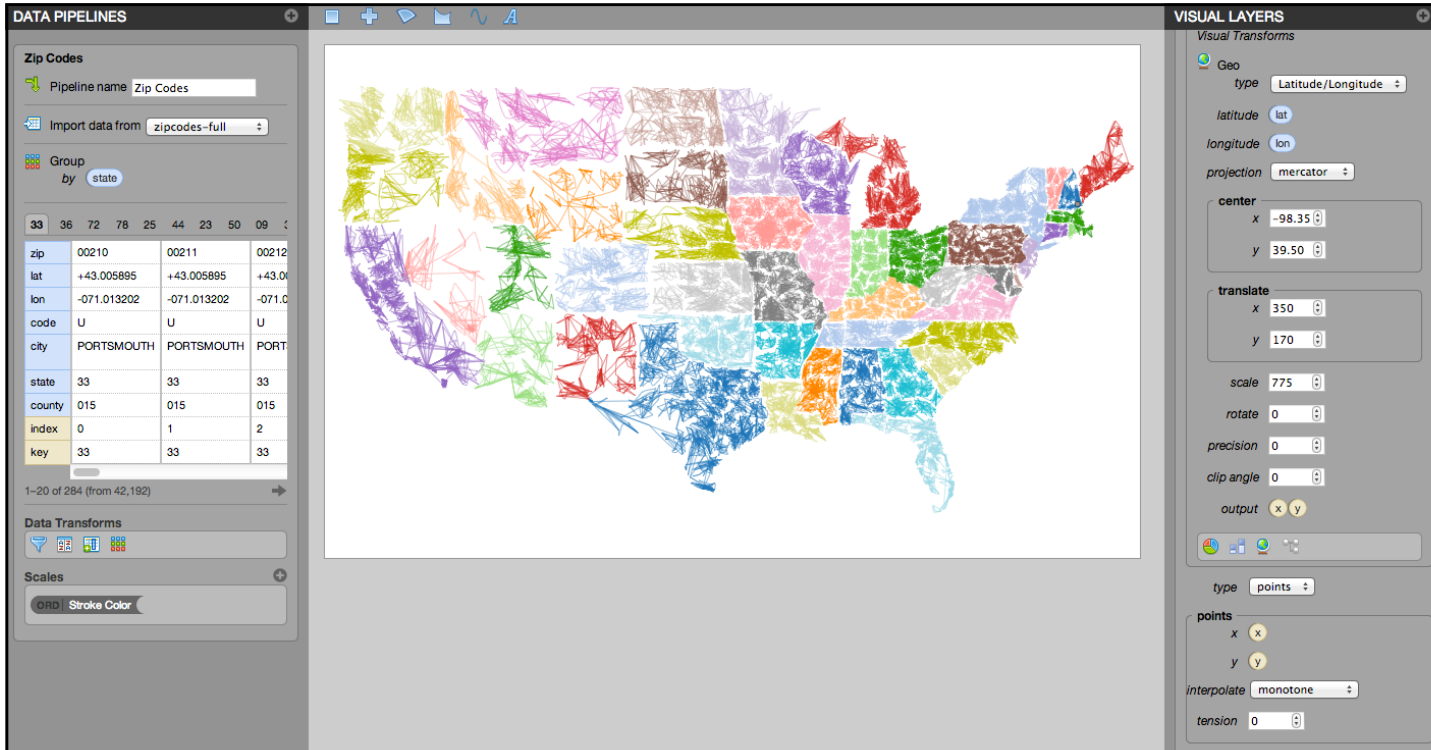
by William Playfair

Lyra A Visualization Design Environment

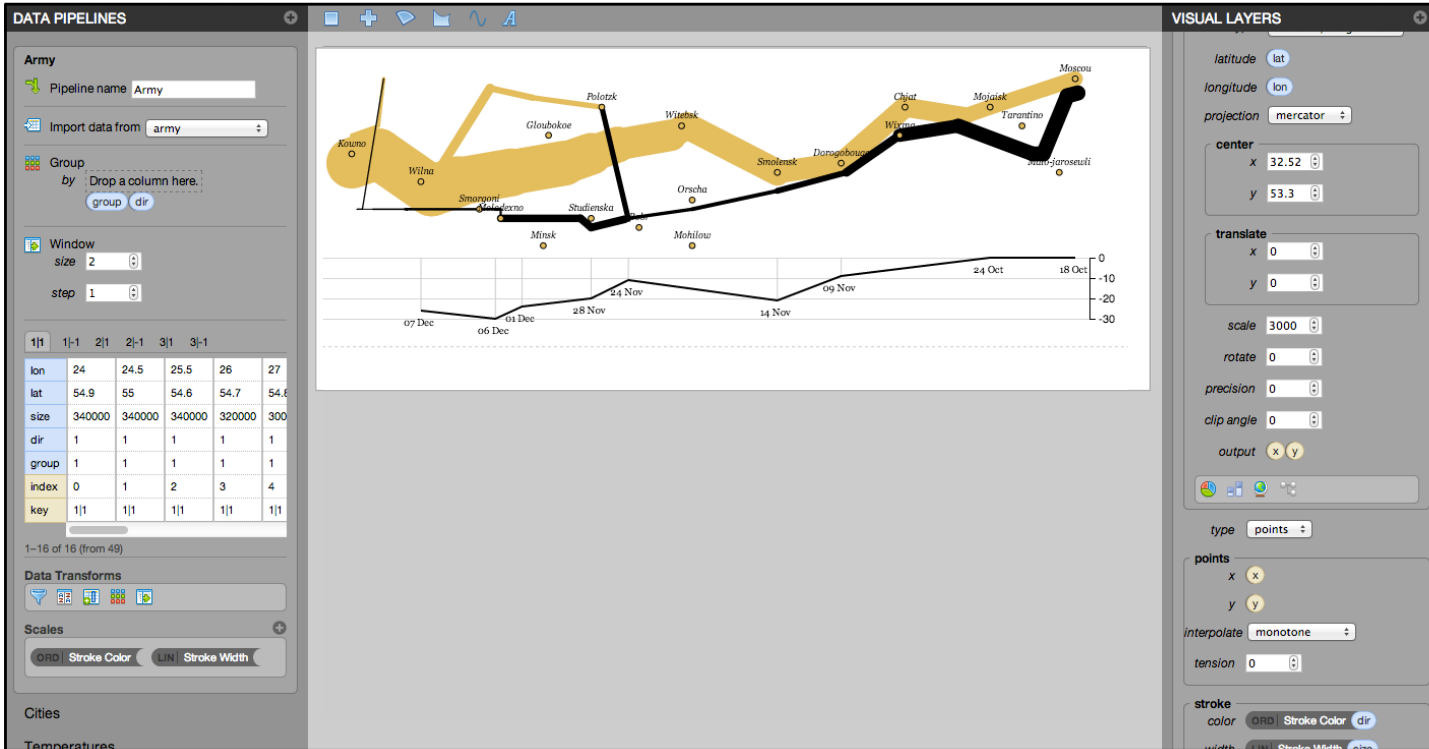


based on the **Railway Timetable** by E. J. Marey

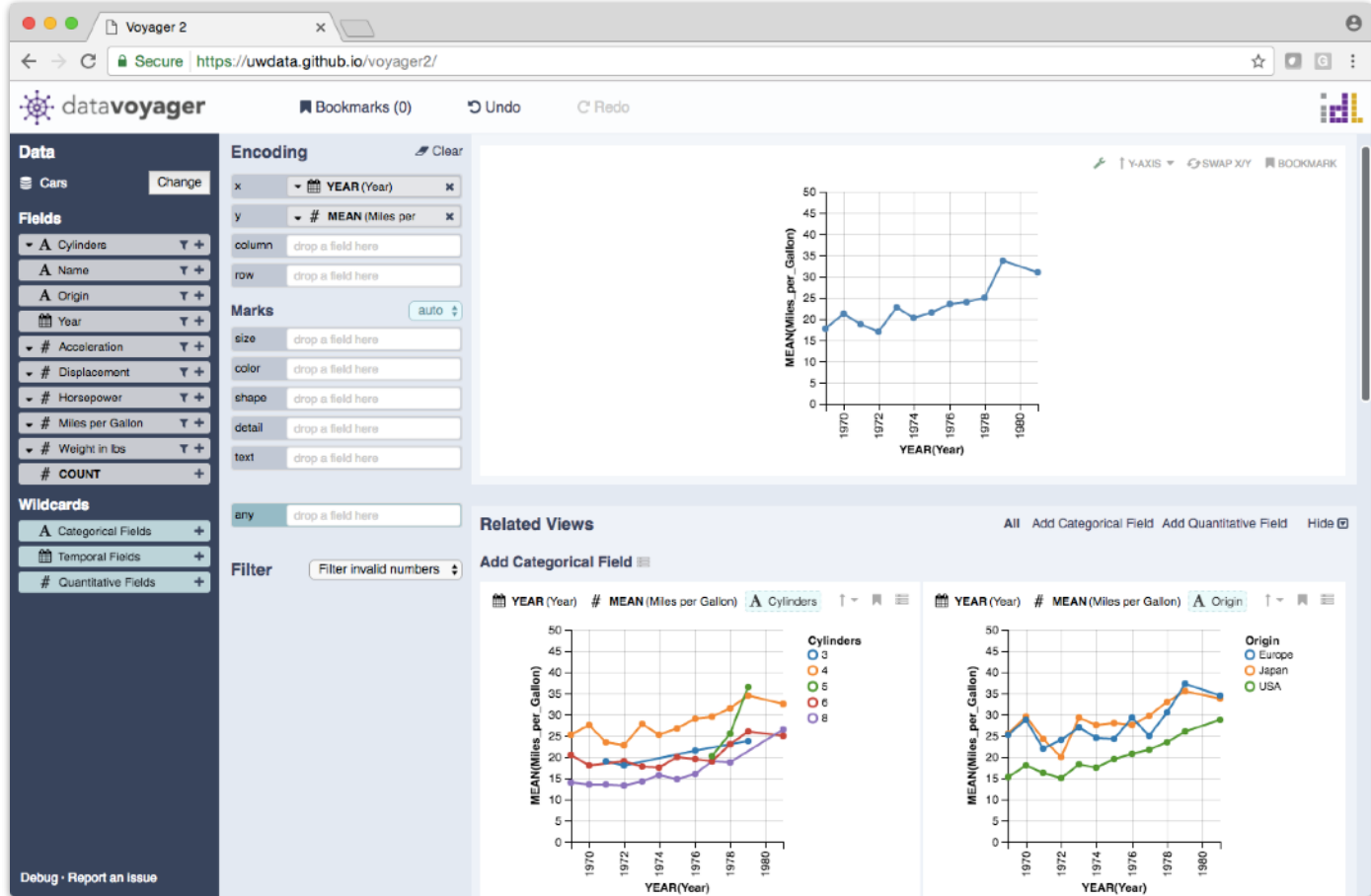
Lyra A Visualization Design Environment



Lyra A Visualization Design Environment



Napoleon's March by Charles Minard



Voyager. Wongsuphasawat et al. *InfoVis'15, CHI'17*

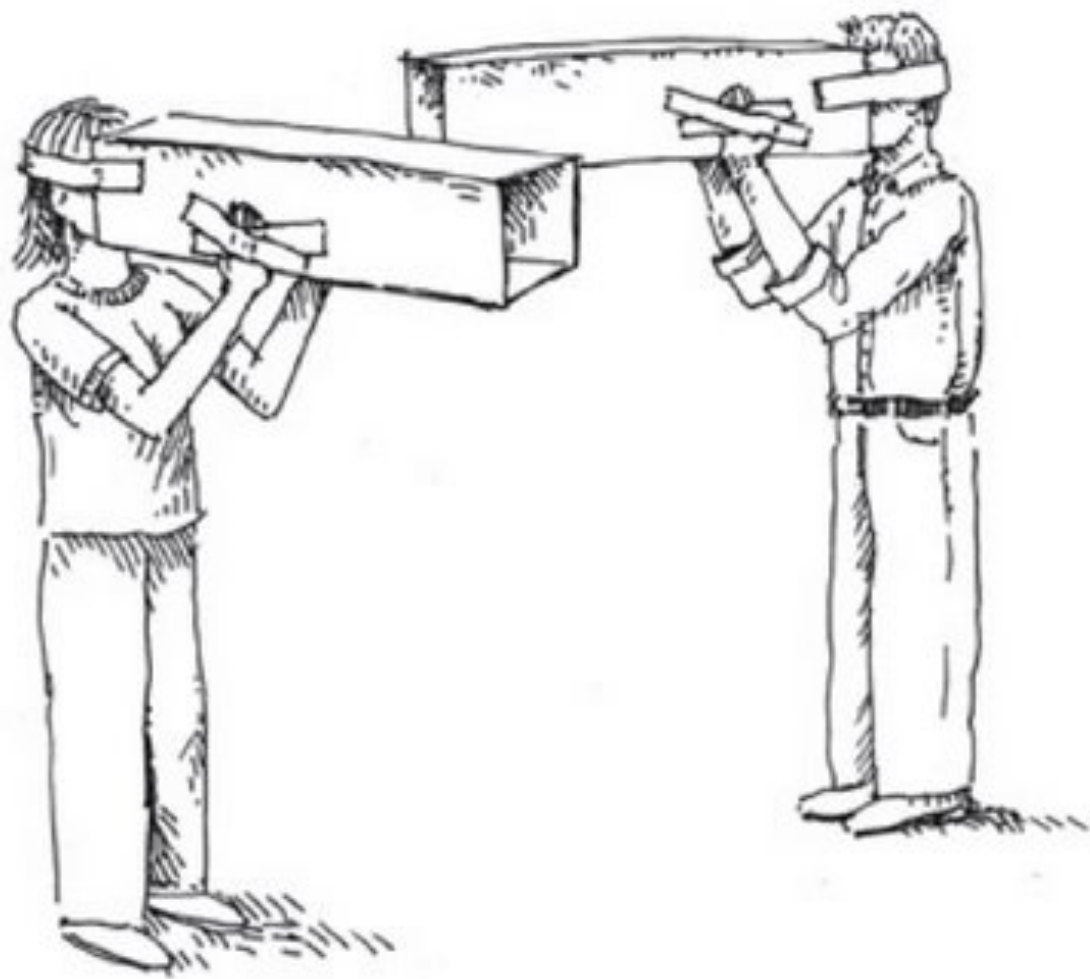
Common exploration pitfalls:

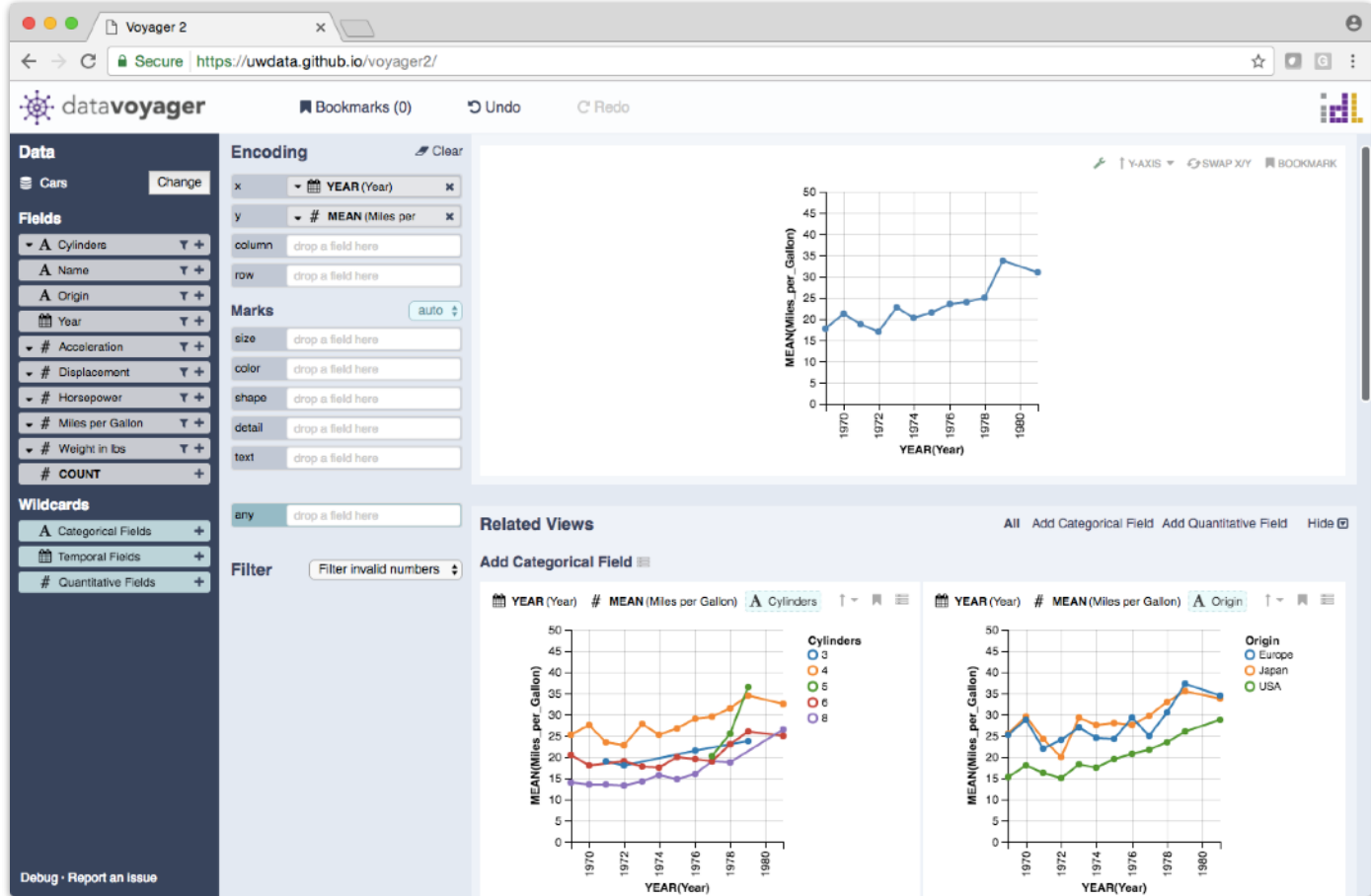
Overlook data quality issues

Fixate on specific relationships

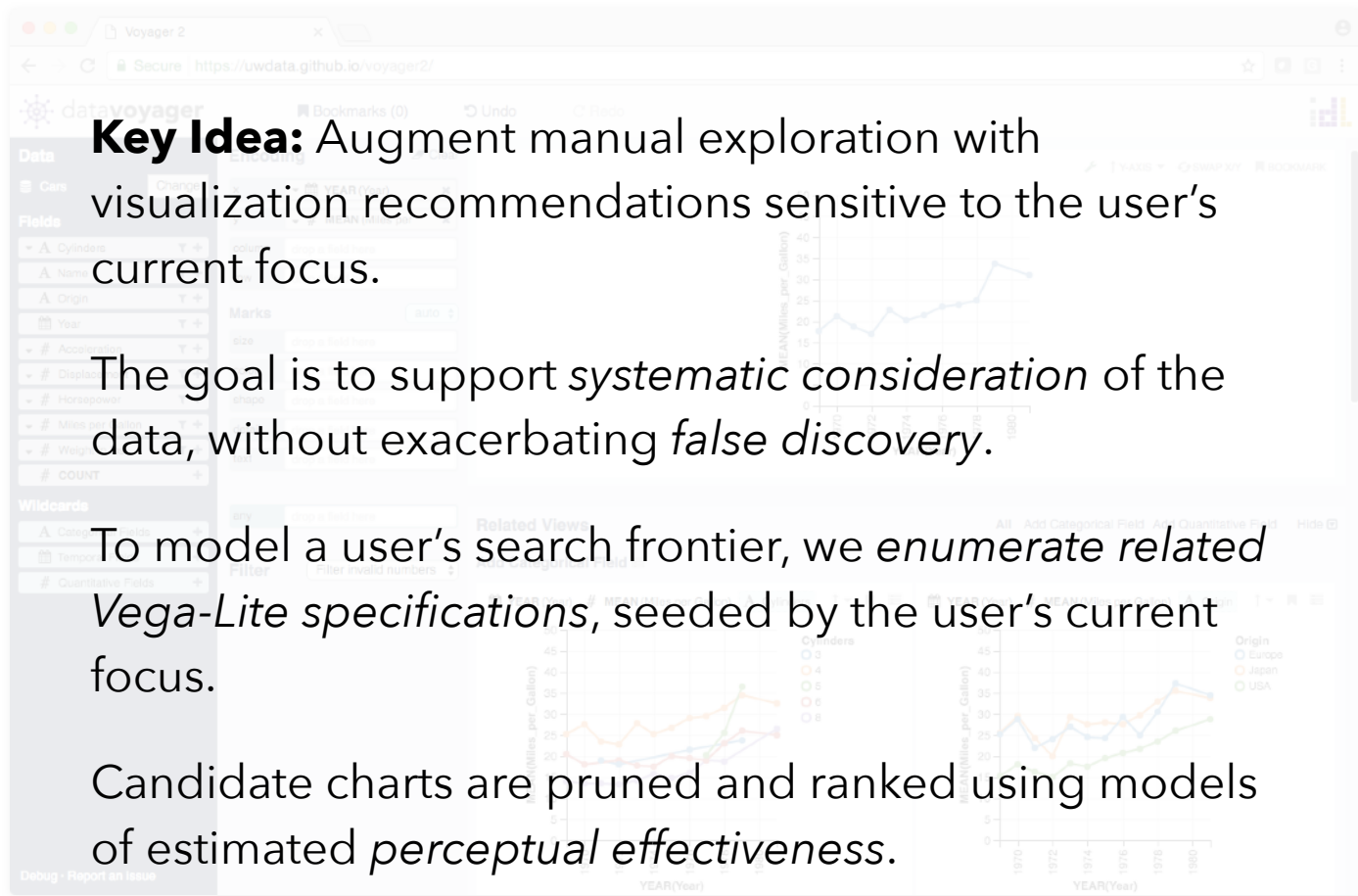
Plus many other biases...

[Heuer 1999, Kahneman 2011, ...]

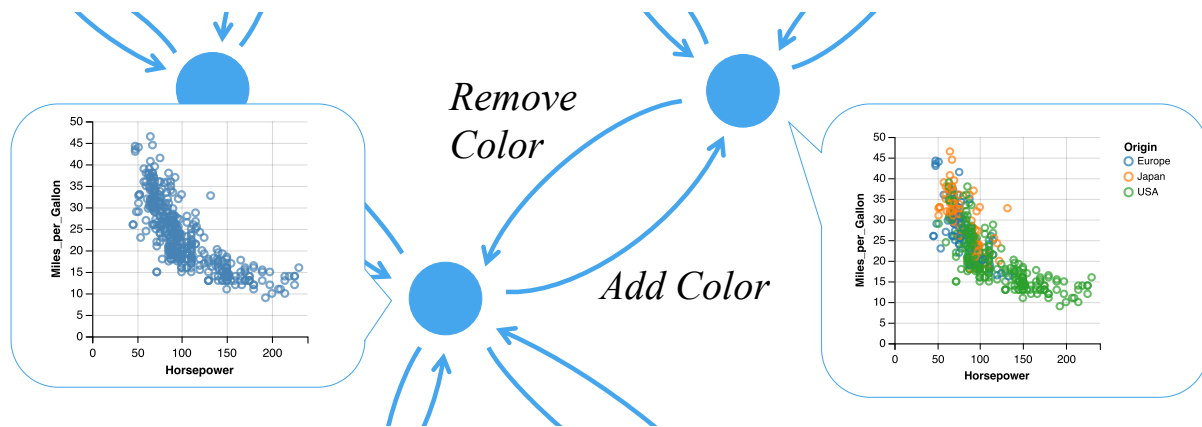




Voyager. Wongsuphasawat et al. *InfoVis'15, CHI'17*

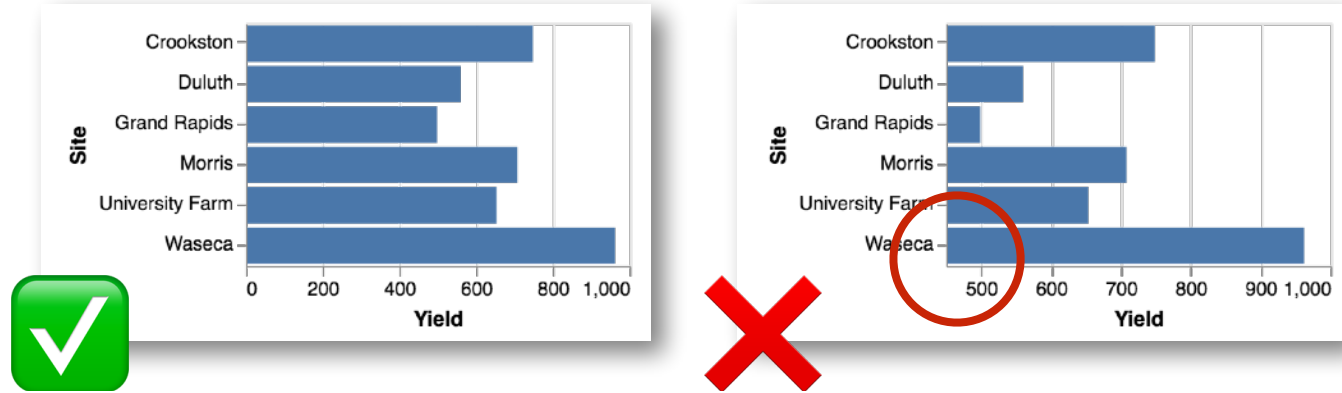


A Formal Design Space of Visualizations



Enumerate Vega-Lite specifications and transformations among them. Search the space using logic programming methods.

Articulate Design Constraints



"Quantitative axes should include a zero baseline"

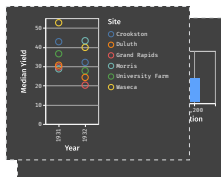
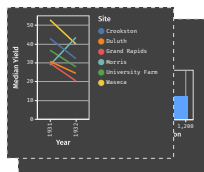
When and how strongly should we apply this?

How to balance with other such constraints?

Learn Design Trade-Offs from Data

Training Data

Pairs of Ranked
Visualizations



Features
Violations of
Design Constraints



Positive
example
 $[u_1, u_2, \dots, u_k]$



Negative
example
 $[v_1, v_2, \dots, v_k]$



Learning Algorithm

Learning to Rank
with Linear SVM

w is the weight
vector of the soft
constraints

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

i : the number of violations
of constraint . i

Compared to other tools, **over 4x more variable sets seen**, and **over 2x more interacted with**.

"related view suggestion accelerates exploration a lot."

"I like that it shows me what fields to include in order to see a specific graph. Otherwise, I have to do a lot of trial and error and can't express what I wanted to see."

"These related views are so good but it's also spoiling that I start thinking less. I'm not sure if that's really a good thing."

Interactive Data Exploration

Tableau, *Lyra*, *Voyager*

Graphical
Interfaces

Visual Analysis Grammars

VizQL, ggplot2, *Vega-Lite*

Declarative
Languages

Visualization Grammars

D3.js, *Vega*

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Processing, OpenGL, Java2D