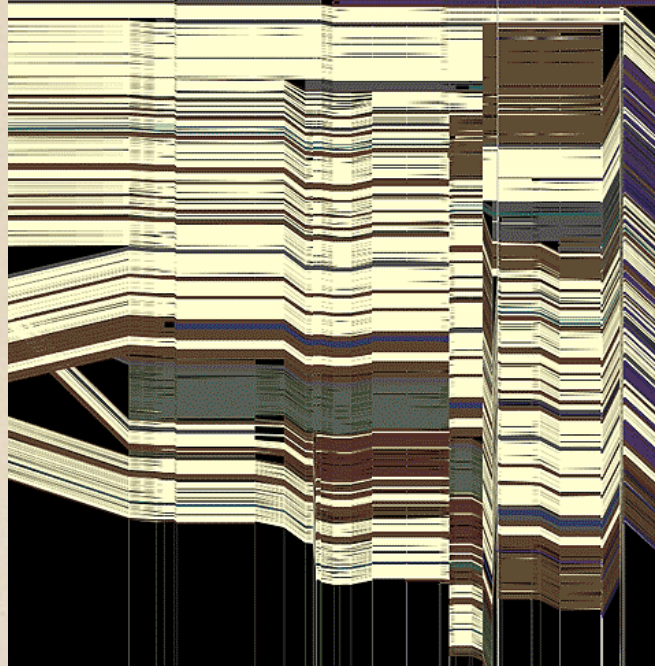
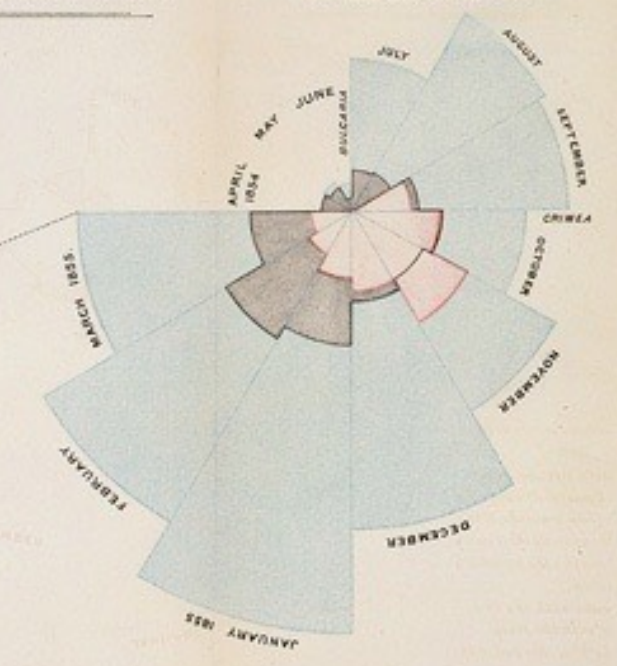


CSE 512 - Data Visualization

Visualization Tools



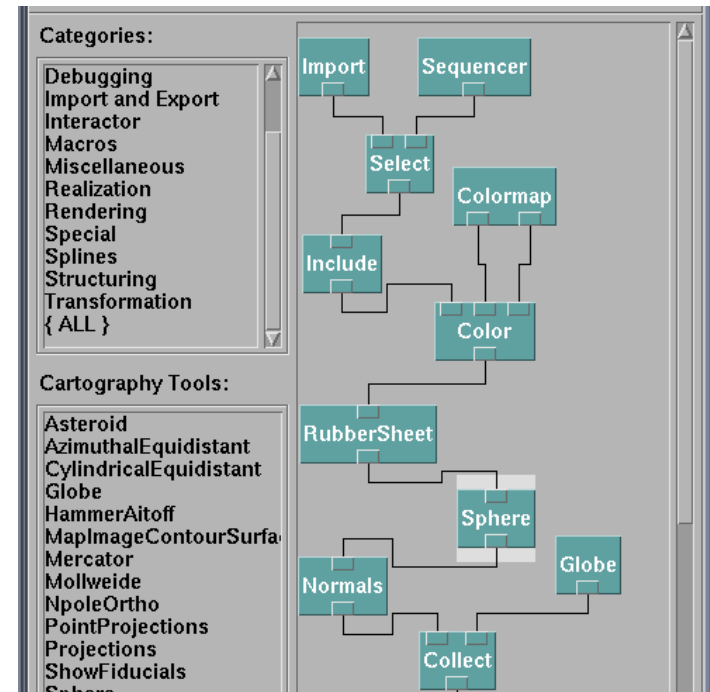
Jeffrey Heer University of Washington

How do people create visualizations?



Chart Typology

Pick from a stock of templates
Easy-to-use but limited expressiveness
Prohibits novel designs, new data types



Component Architecture

Permits more combinatorial possibilities
Novel views require new operators,
which requires software engineering

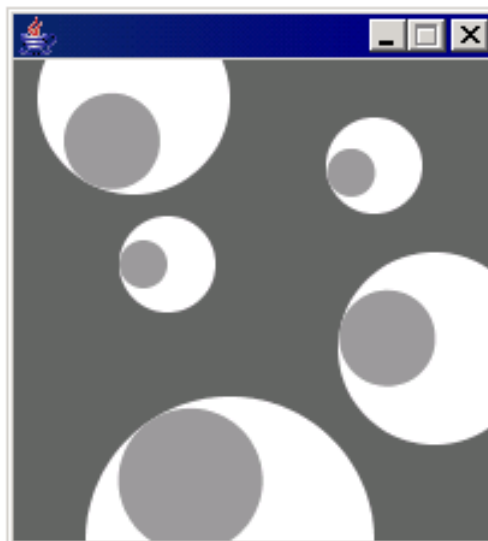
Graphics APIs

Processing, OpenGL, Java2D



sketch_070126a \$

```
    ey = y;  
    size = s;  
  }  
  
  void update(int mx, int my) {  
    angle = atan2(my-ey, mx-ex);  
  }  
  
  void display() {  
    pushMatrix();  
    translate(ex, ey);  
    fill(255);  
    ellipse(0, 0, size, size);  
    rotate(angle);  
    fill(153);  
    ellipse(size/4, 0, size/2, size/2);  
    popMatrix();  
  }  
}
```





US Air Traffic, Aaron Koblin

Graphics APIs

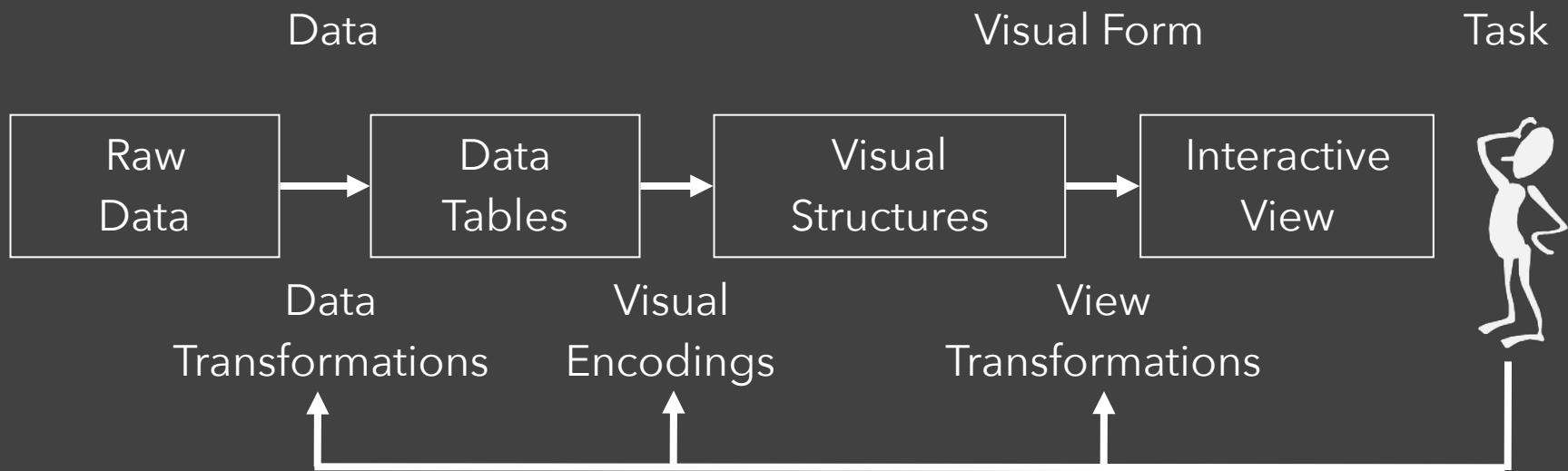
Processing, OpenGL, Java2D

Component Architectures

Prefuse, Flare, Improvise, VTK

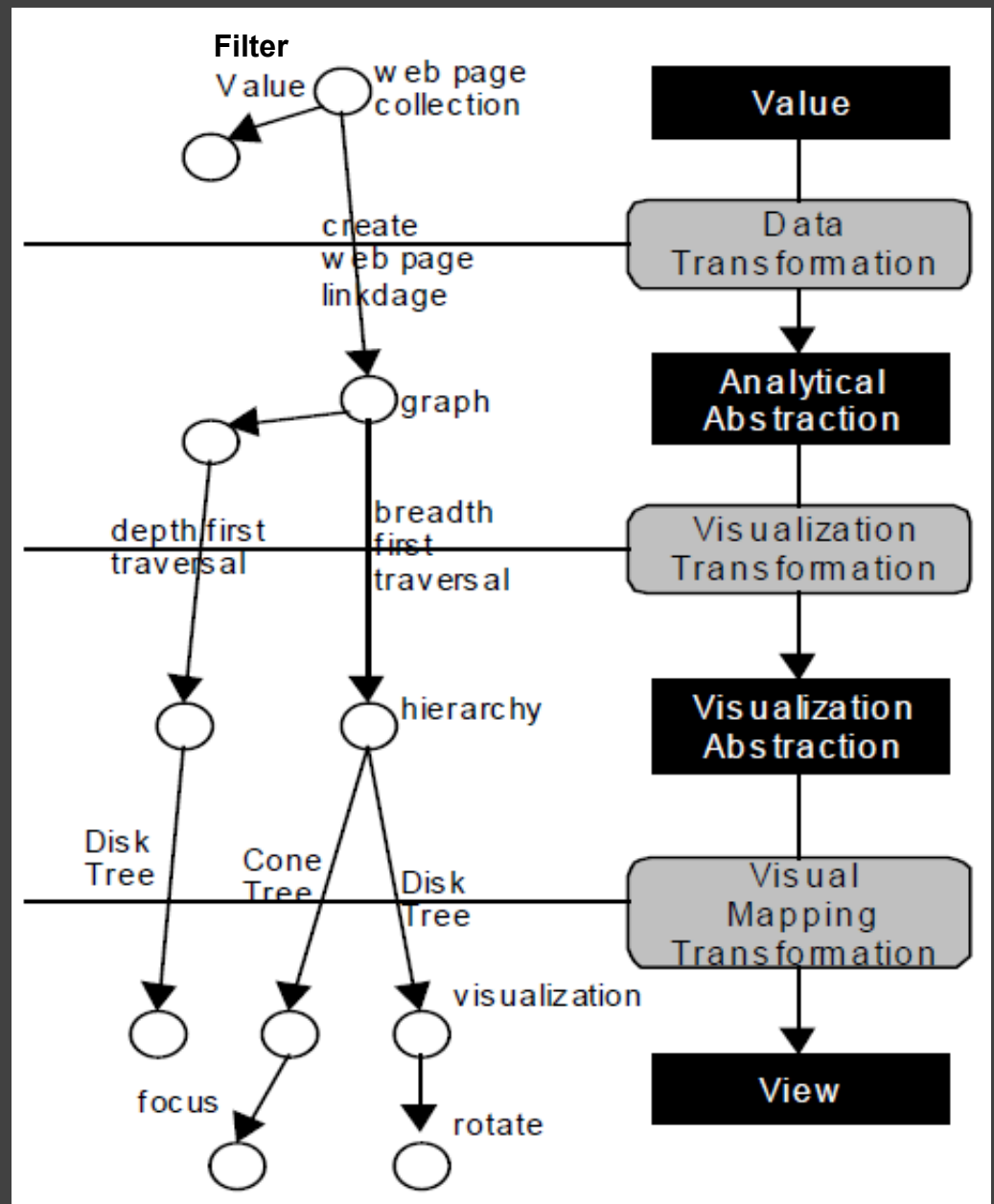
Graphics APIs

Processing, OpenGL, Java2D



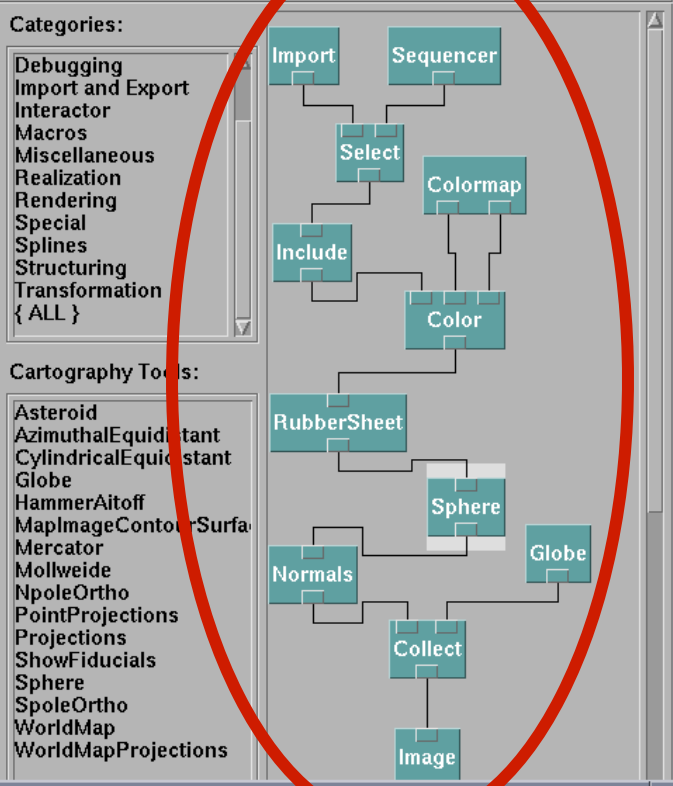
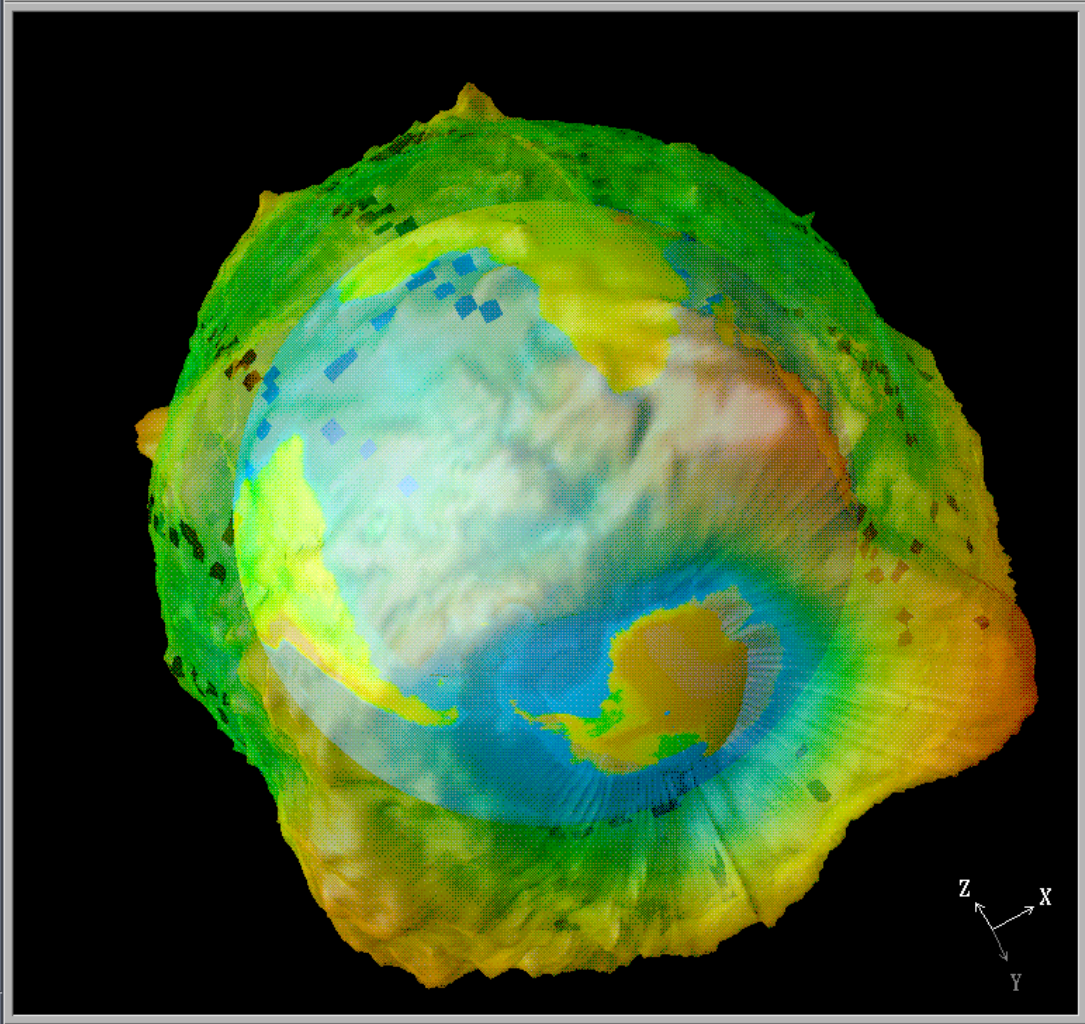
Data State Model

[Chi 98]



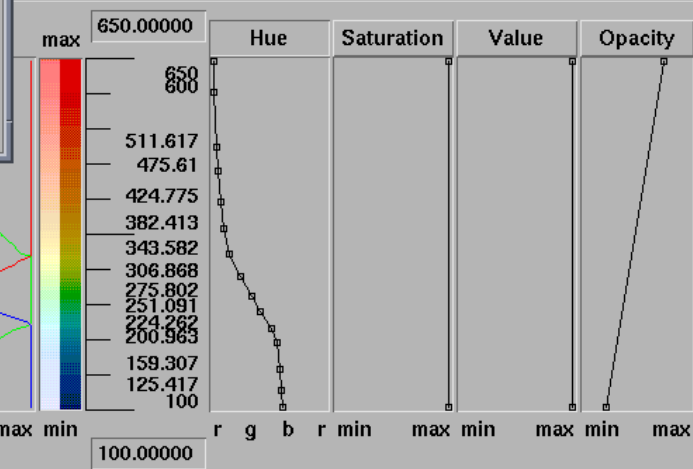
File Execute Windows Connection Options Help

File Edit Execute Windows Connection Options Help



Colormap Editor

File Execute Options Help



View Control...

Undo Ctrl+U Redo Ctrl+D

Mode: Rotate

Set View: None

Projection: Perspective

View Angle: 30.000

Close Reset Ctrl+F

Sequence Control

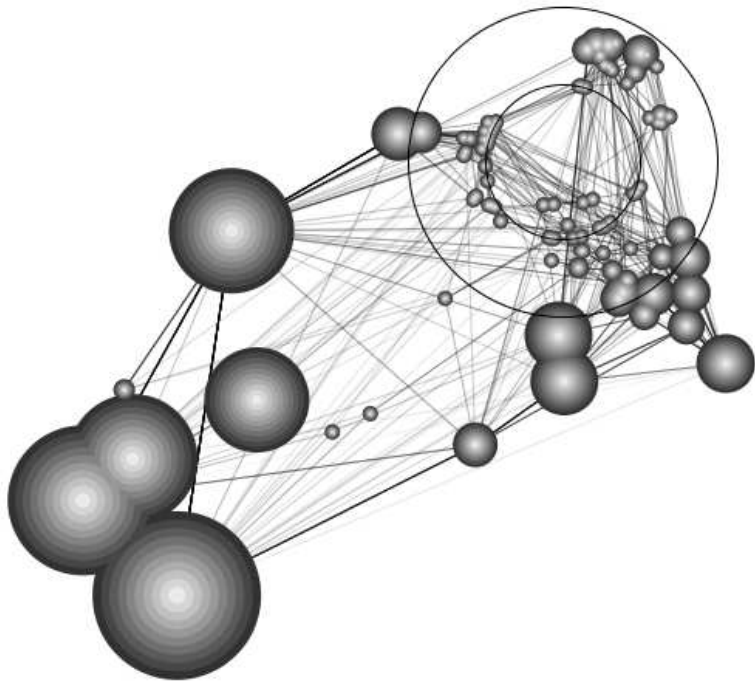
⏪ ⏩ ⏸ ⏹

⏮ ⏭ ⏪ ⏩ ⏸ ⏹

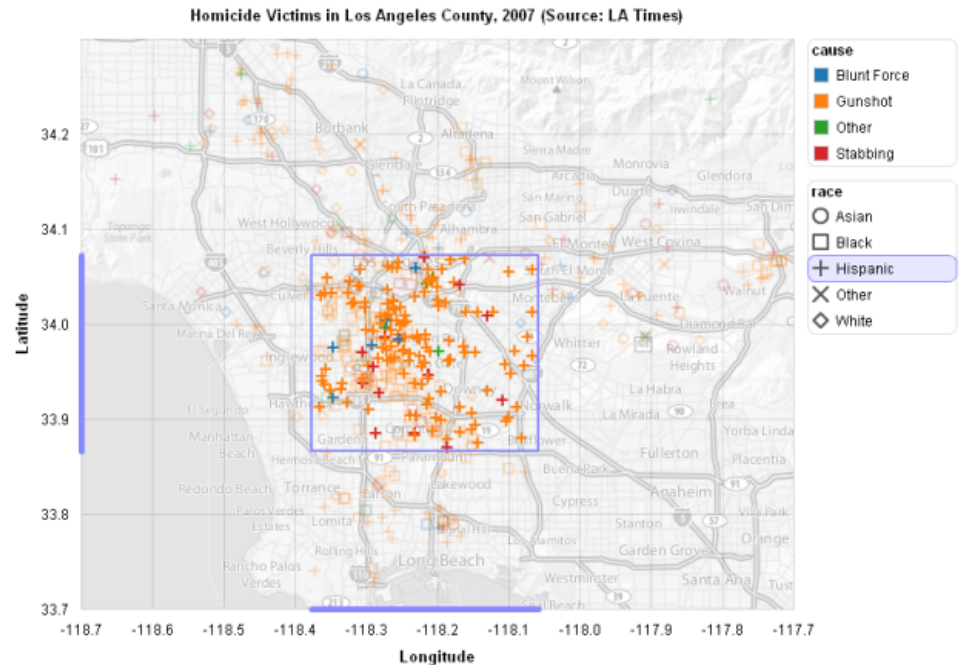
Prefuse & Flare

Operator-based toolkits for visualization design

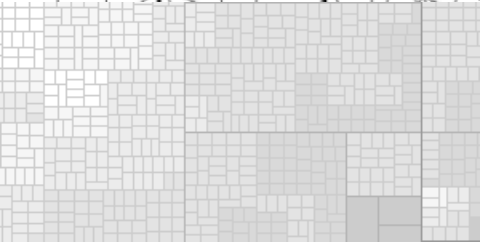
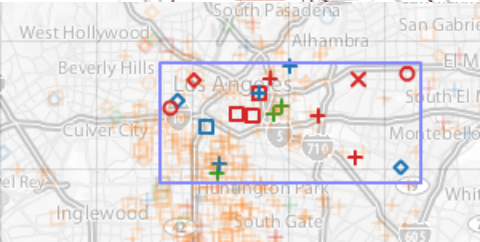
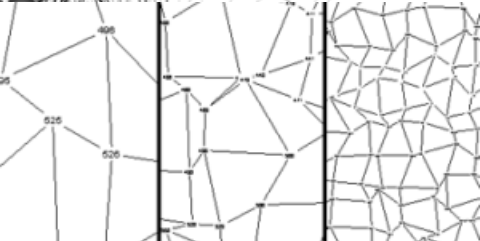
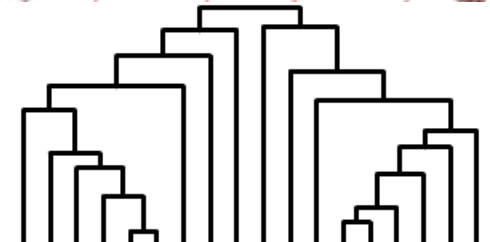
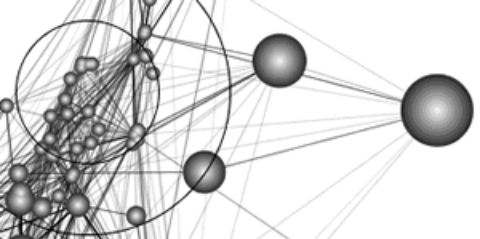
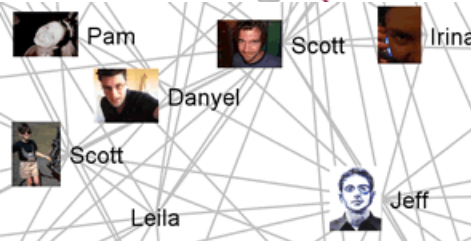
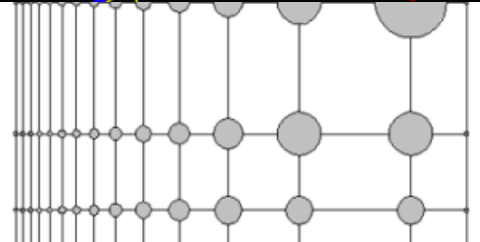
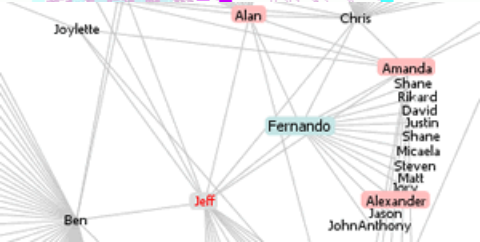
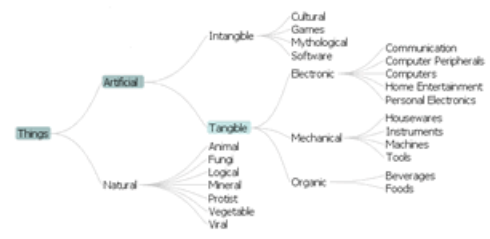
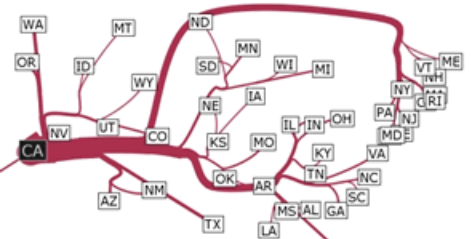
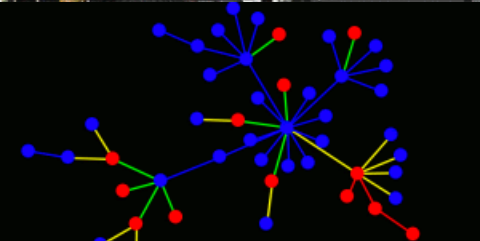
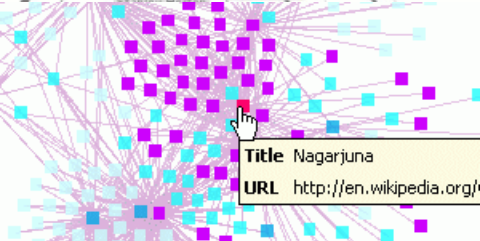
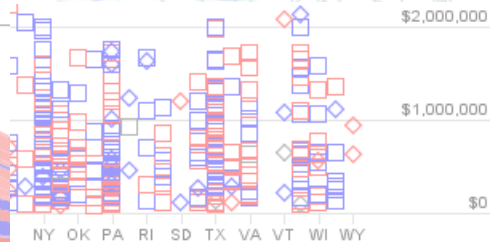
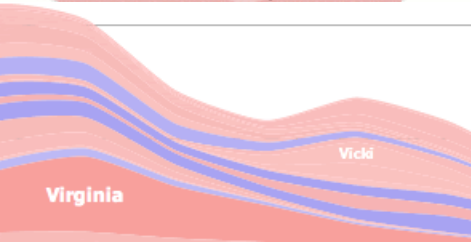
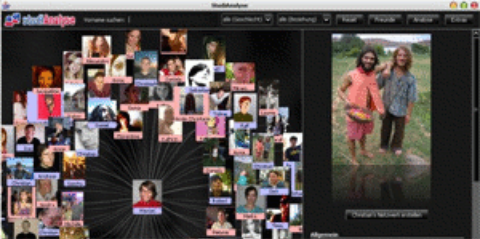
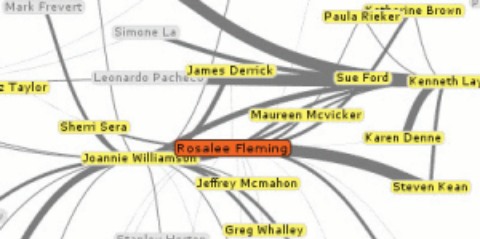
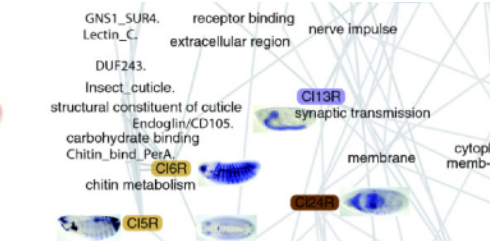
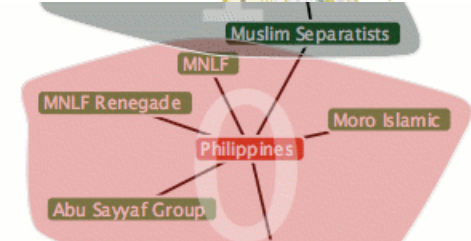
Vis = (Input Data -> Visual Objects) + Operators



Prefuse (<http://prefuse.org>)



Flare (<http://flare.prefuse.org>)





?

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Chart Typologies

Excel, Many Eyes, Google Charts

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D



Chart Typologies

Data Sets : State Quick Facts

Uploaded By: [zinggoat](#)

Created at: Friday May 18, 3:08 PM

Data Source: [US Census Bureau](#)

Description:

Tags: [people](#) [census](#)

[view as text](#)

[edit data set](#)

	People QuickFacts	Population 2005 estimate	Population percent change April 1 2000 to July 1 2005	Population 2000	Population percent change 1990 to 2000	Persons under 5 years old percent 2004	Persons under 18 years old percent 2004	Persons 65 years old and over percent 2004
1	Alabama	4557808	0.03	4447100	0.1	0.07	0.24	0.13
2	Alaska	663661	0.06	626932	0.14	0.08	0.29	0.06
3	Arizona	5939292	0.16	5130632	0.4	0.08	0.27	0.13
4	Arkansas	2779154	0.04	2673400	0.14	0.07	0.25	0.14
5	California	36132147	0.07	33871648	0.14	0.07	0.27	0.11
6	Colorado	4665177	0.08	4301261	0.31	0.07	0.26	0.1
7	Connecticut	3510297	0.03	3405565	0.04	0.06	0.24	0.14
8	Delaware	843524	0.08	783600	0.18	0.07	0.23	0.13
9	Florida	17789864	0.11	15982378	0.24	0.06	0.23	0.17
10	Georgia	9072576	0.11	8186453	0.26	0.08	0.26	0.1
11	Hawaii	1275194	0.05	1211537	0.09	0.07	0.24	0.14
12	Idaho	1429096	0.1	1293953	0.29	0.07	0.27	0.11
13	Illinois	12763371	0.03	12419293	0.09	0.07	0.26	0.12



Choosing a visualization type for **State Quick Facts**

Analyze a text



Tag Cloud

How are you using your words? This enhanced tag cloud will show you the words popularity in the given set of text.

[Learn more](#)



Wordle

Wordle is a toy for generating "word clouds" from text that you provide. The clouds give greater prominence to words that appear more frequently in the source text.

[Learn more](#)

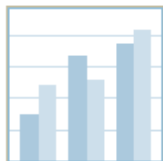


Word Tree

See a branching view of how a word or phrase is used in a text. Navigate the text by zooming and clicking.

[Learn more](#)

Compare a set of values



Bar Chart

How do the items in your data set stack up? A bar chart is a simple and recognizable way to compare values. You can display several sets of bars for multivariate comparisons.

[Learn more](#)



Block Histogram

This versatile chart lets you get a quick sense of how a single set of data is distributed. Each item in the data is an individually identifiable block.

[Learn more](#)

Visualizations : Federal Spending by State, 2004

Creator: Anonymous

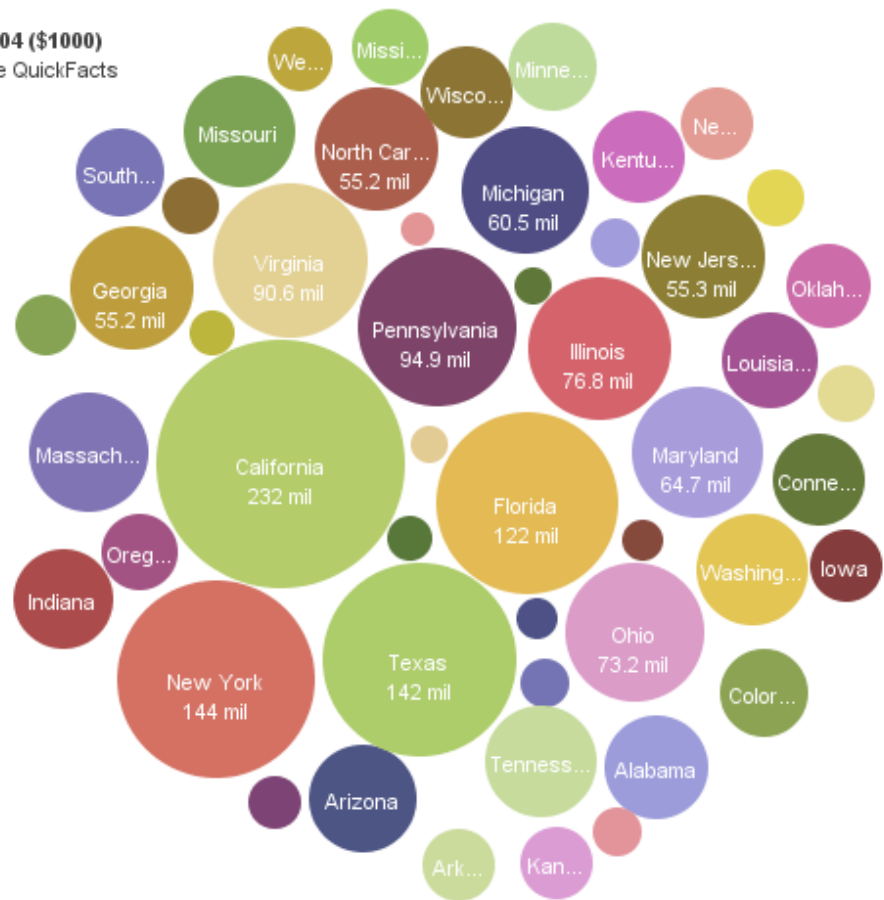
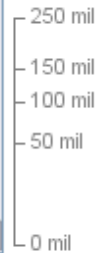
Tags: census people

People QuickFac...

Click to select,
Ctrl-Click: multiple
Shift-Click: range

Federal spending 2004 (\$1000)
Disks colored by People QuickFacts

- Alabama
- Alaska
- Arizona
- Arkansas
- California
- Colorado
- Connecticut
- Delaware
- Florida
- Georgia
- Hawaii
- Idaho
- Illinois
- Indiana
- Iowa
- Kansas
- Kentucky
- Louisiana
- Maine
- Maryland



Search>>

To highlight or find totals
click or ctrl-click.

Bubble Size: Federal spending 2004 (\$1000) | Label: People QuickFacts | Color: People QuickFacts

- Retail sales per capita 2002
- Minority-owned firms percent of total 1997
- Women-owned firms percent of total 1997
- Housing units authorized by building permits 2004
- Federal spending 2004 (\$1000)
- Land area 2000 (square miles)
- Persons per square mile 2000
- FIPS Code

Census Bureau This data set has not yet been rated

Comments (1)



MAD LIBS®

MY MUSIC LESSON

Every Wednesday, when I get home from school, I have a piano lesson. My teacher is a very strict house
NOUN. Her name is Hillary Clinton
CELEBRITY (FEMALE). Our piano is a Steinway Concert tree
NOUN and it has 88 ~~keys~~ cups
PLURAL NOUN. It also has a soft pedal and a/an Smily
ADJECTIVE pedal. When I have a lesson, I sit down on the piano AIBERTO
NOUN and play for 16 minutes
PERIOD OF TIME. I do scales to exercise my cats
PLURAL NOUN, and then I usually play a minuet by Johann Sebastian washington
CELEBRITY (LAST NAME). Teacher says I am a natural Haunted House
NOUN and have a good musical leg
PART OF THE BODY. Perhaps when I get better I will become a concert vet
PROFESSION and give a recital at Carnegie hospital
TYPE OF BUILDING.

[M]ost charting packages channel user requests into a **rigid array of chart types**. To atone for this lack of flexibility, they offer a kit of post-creation editing tools to return the image to what the user originally envisioned. **They give the user an impression of having explored data rather than the experience.**

Leland Wilkinson
The Grammar of Graphics, 1999

Chart Typologies

Excel, Many Eyes, Google Charts

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D



Schema ×

congress.csv Connection

Find:

Dimensions

- Abc Candidate
- Abc Candidate ID
- Abc General Elec Status
- Abc Incumbent/Challenger/Open-Seal
- # Party
- Abc Party Desig
- Abc Primary Elec Status
- Abc Runoff Elec Status
- Abc Spec Elec Status
- Abc State Code
- # Year
- Abc Measure Names

Measures

- # District
- # General Elec Pct
- # Total Receipts
- # Measure Values

Groups

Columns: Party Year

Rows: SUM(Total..)

Filters:

Level of Detail:

Mark: Automatic

Text:

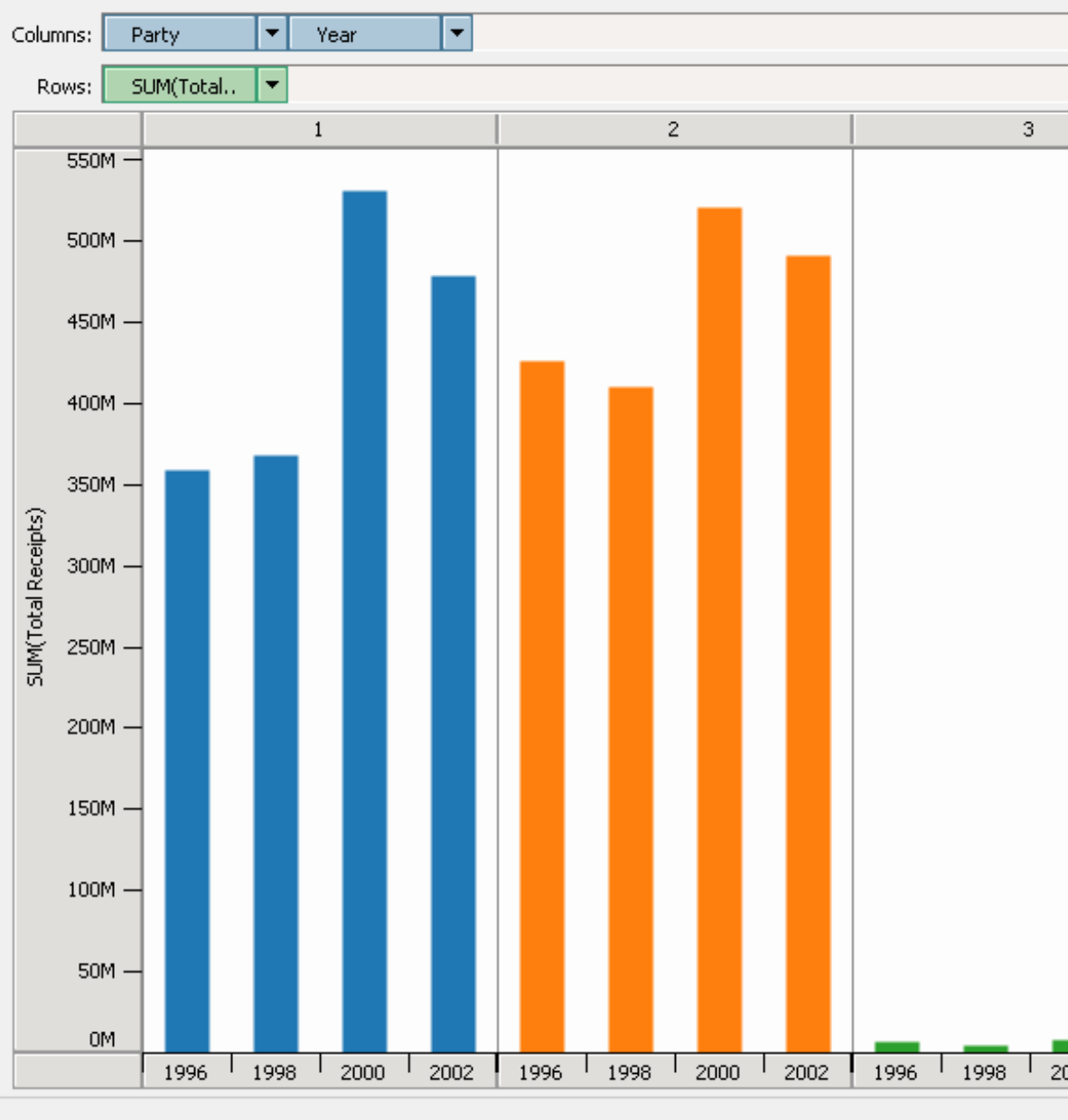
Color: Party

Size:

Legend:

- 1
- 2
- 3

Size:



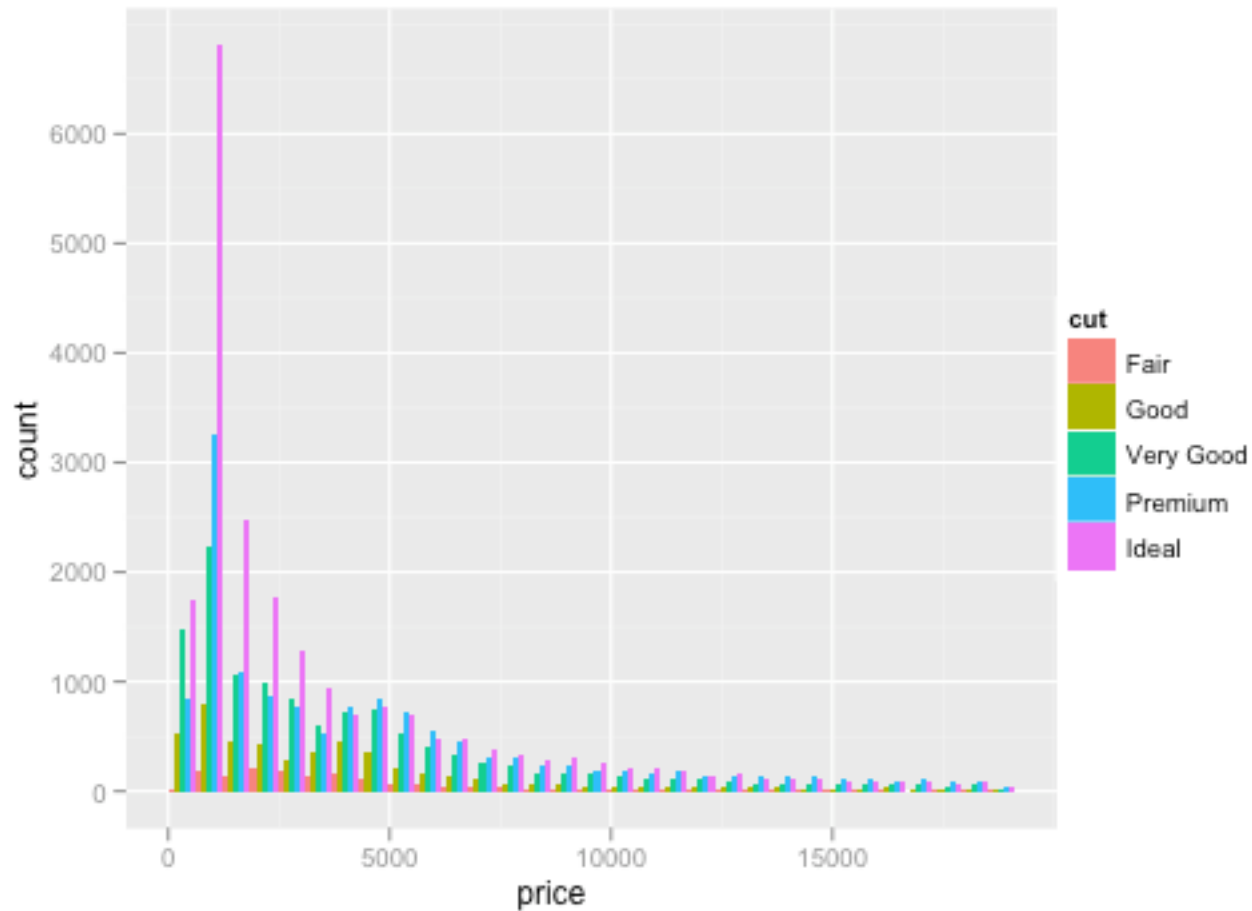
Statistics and Computing

Leland Wilkinson

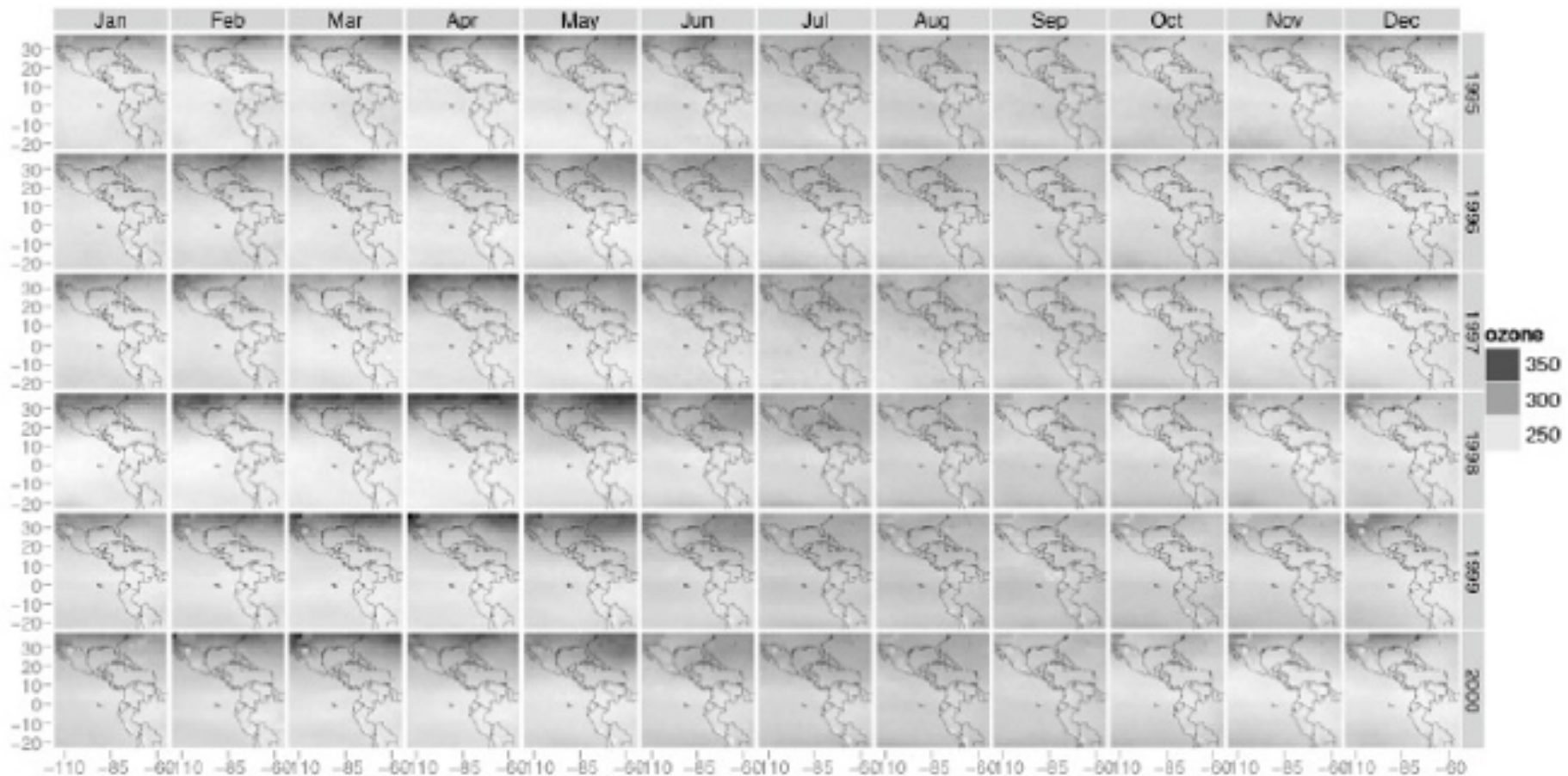
**The Grammar
of Graphics**

Second Edition

 Springer



```
ggplot(diamonds, aes(x=price, fill=cut))  
+ geom_bar(position="dodge")
```



```

qplot(long, lat, data = expo, geom = "tile", fill = ozone,
      facets = year ~ month) +
scale_fill_gradient(low = "white", high = "black") + map

```

Ease-of-Use



Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2

?

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness



Ease-of-Use



Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2

Visualization Grammars

Protovis, D3.js

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

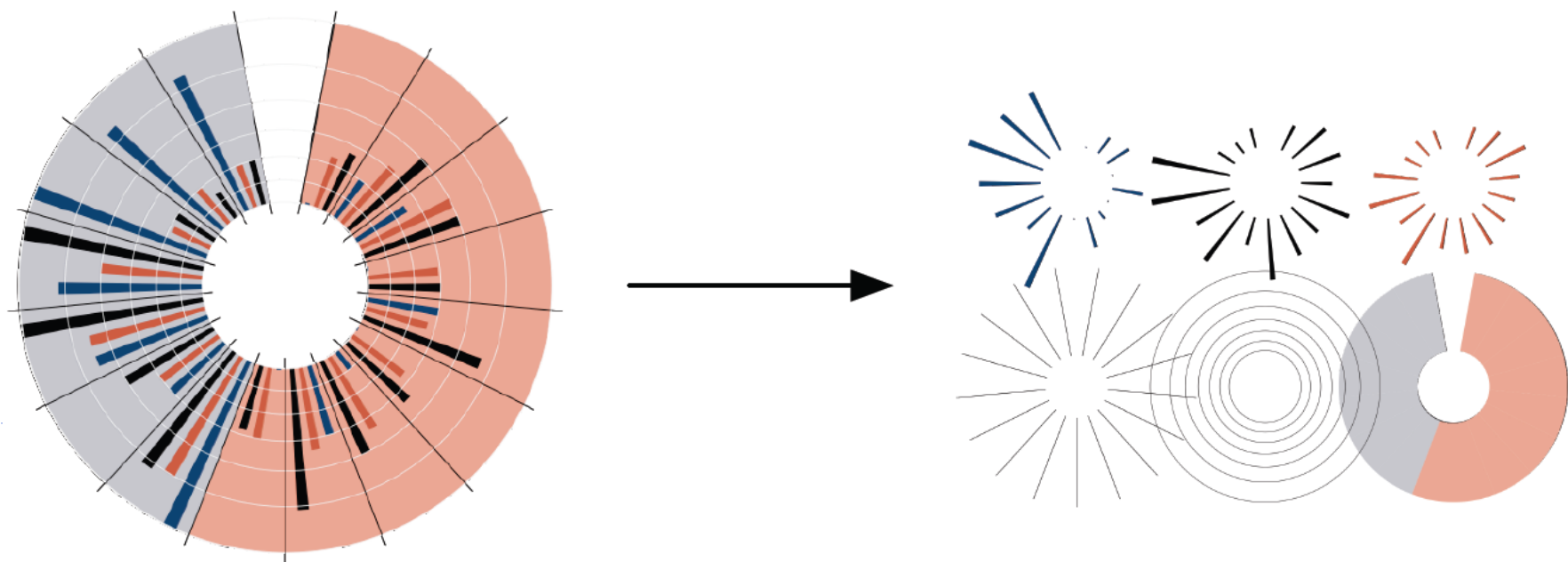


Protovis & D3

Today's first task is not to invent wholly new [*graphical*] techniques, though these are needed. Rather we need most vitally to recognize and reorganize the **essential of old techniques**, to **make easy their assembly in new ways**, and to **modify their external appearances to fit the new opportunities**.

J. W. Tukey, M. B. Wilk
Data Analysis & Statistics, 1965

Protovis: A Grammar for Visualization

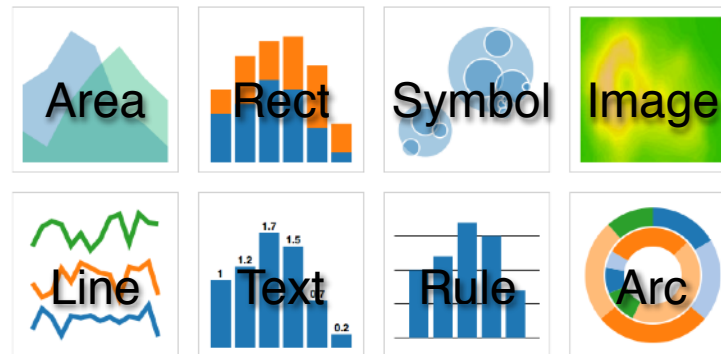


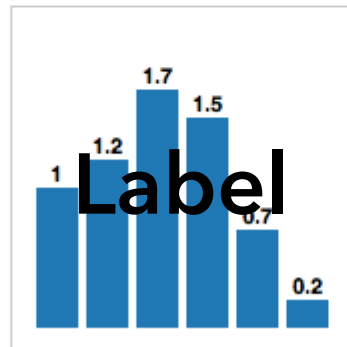
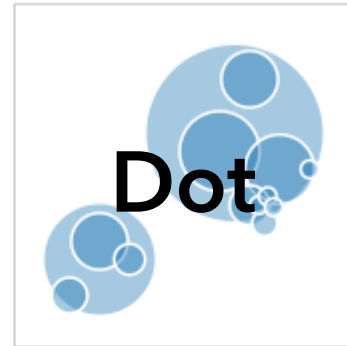
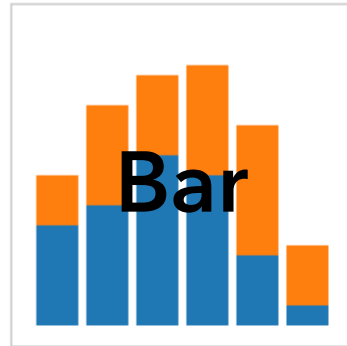
A graphic is a composition of data-representative marks.

with **Mike Bostock** & **Vadim Ogievetsky**

Visualization Grammar

Data	Input data to visualize
Transforms	Grouping, stats, projection, layout
Scales	Map data values to visual values
Guides	Axes & legends visualize scales
Marks	Data-representative graphics



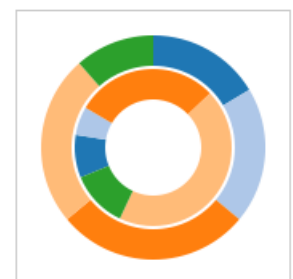
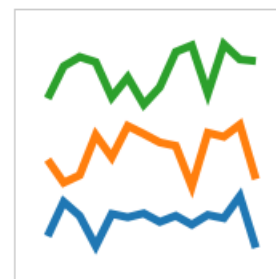
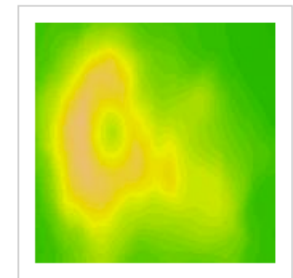
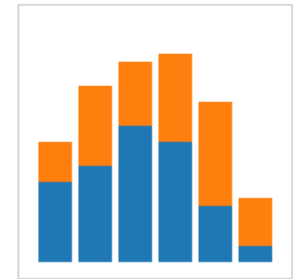


MARKS: Protovis graphical primitives

MARK

 $\lambda : D \rightarrow R$

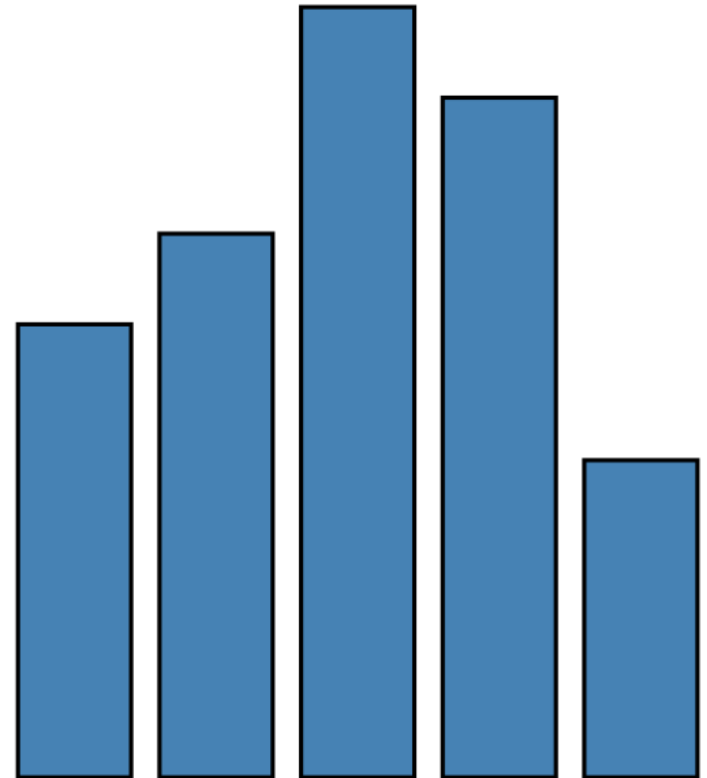
data	λ
visible	λ
left	λ
bottom	λ
width	λ
height	λ
fillStyle	λ
strokeStyle	λ
lineWidth	λ
...	λ



RECT

$\lambda : D \rightarrow R$

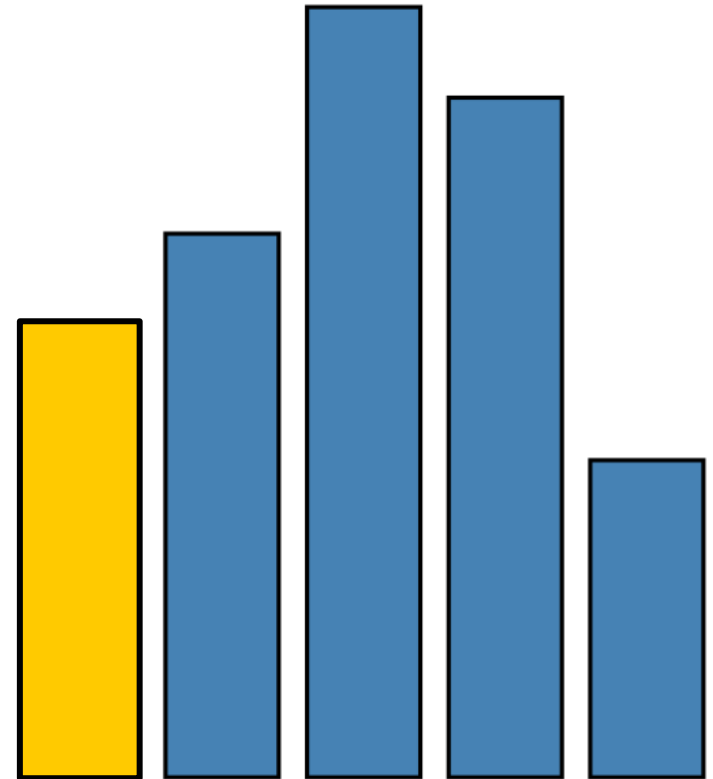
data	1	1.2	1.7	1.5	0.7
visible	true				
left	$\lambda: \text{index} * 25$				
bottom	0				
width	20				
height	$\lambda: \text{datum} * 80$				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



RECT

$\lambda : D \rightarrow R$

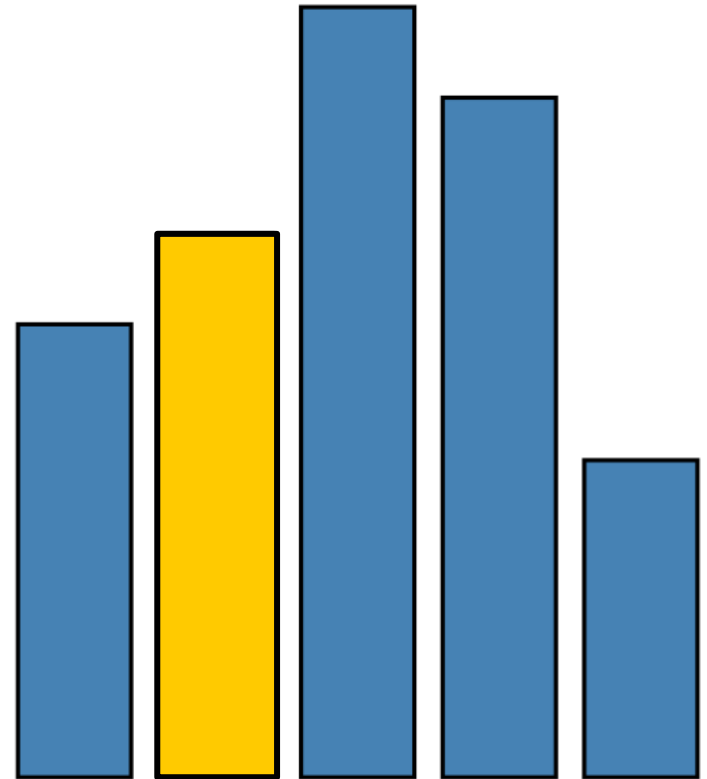
data	1	1.2	1.7	1.5	0.7
visible	true				
left	0 * 25				
bottom	0				
width	20				
height	1 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



RECT

$\lambda : D \rightarrow R$

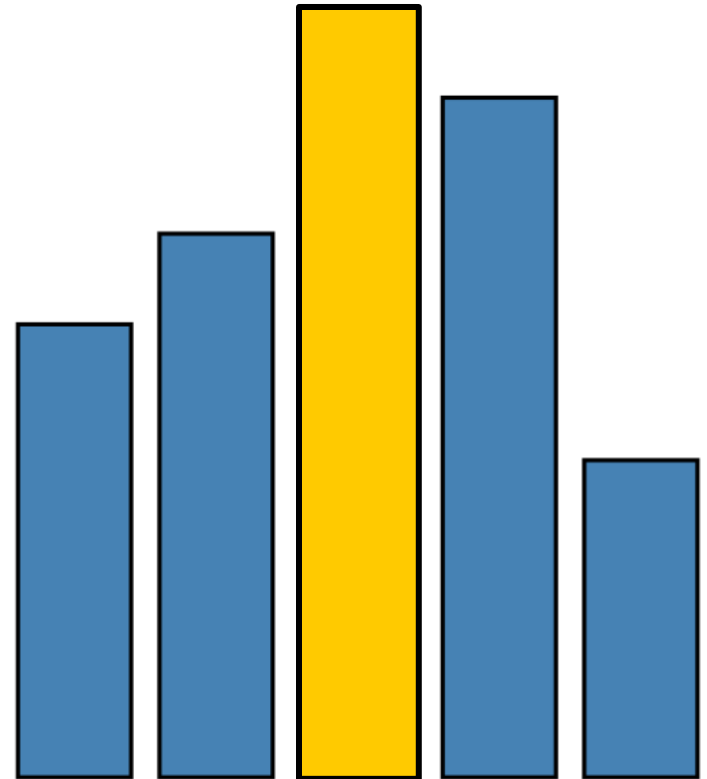
data	1	1.2	1.7	1.5	0.7
visible	true				
left	1 * 25				
bottom	0				
width	20				
height	1.2 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



RECT

$\lambda : D \rightarrow R$

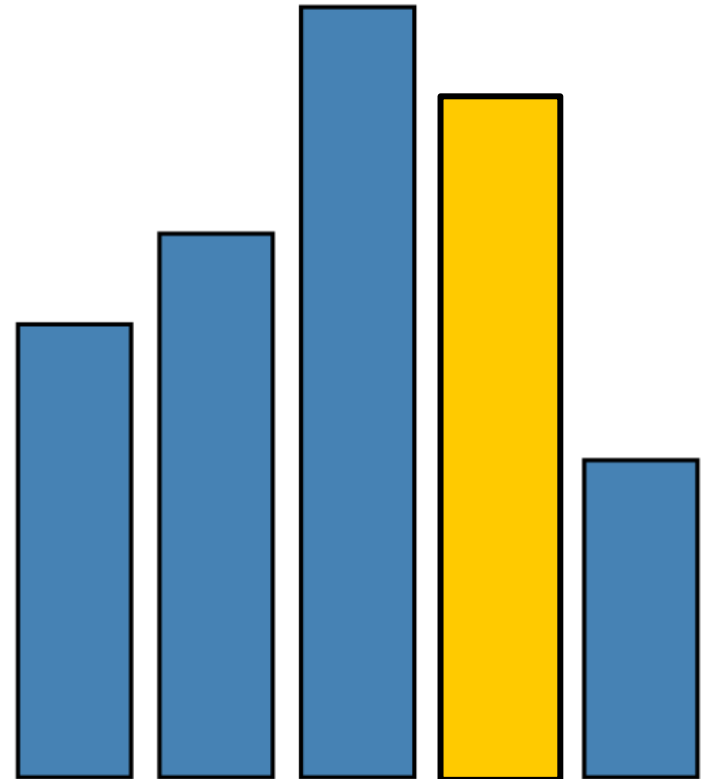
data	1	1.2	1.7	1.5	0.7
visible	true				
left	2 * 25				
bottom	0				
width	20				
height	1.7 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



RECT

$\lambda : D \rightarrow R$

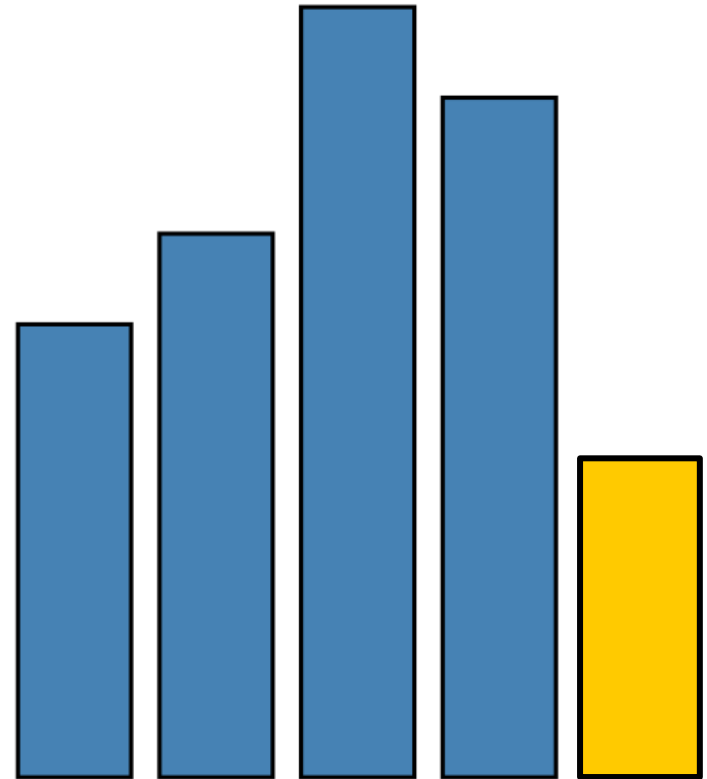
data	1	1.2	1.7	1.5	0.7
visible	true				
left	3 * 25				
bottom	0				
width	20				
height	1.5 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



RECT

$\lambda : D \rightarrow R$

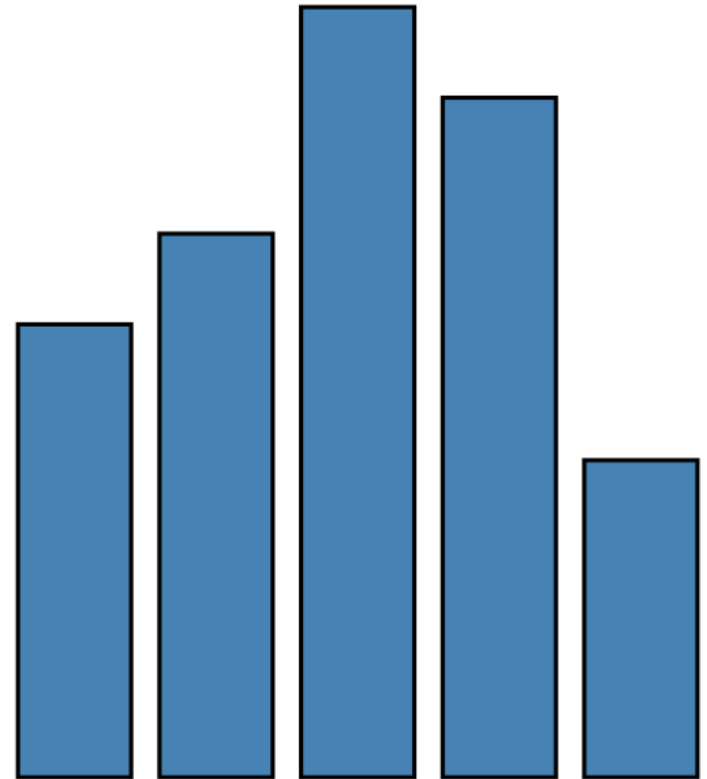
data	1	1.2	1.7	1.5	0.7
visible	true				
left	4 * 25				
bottom	0				
width	20				
height	0.7 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



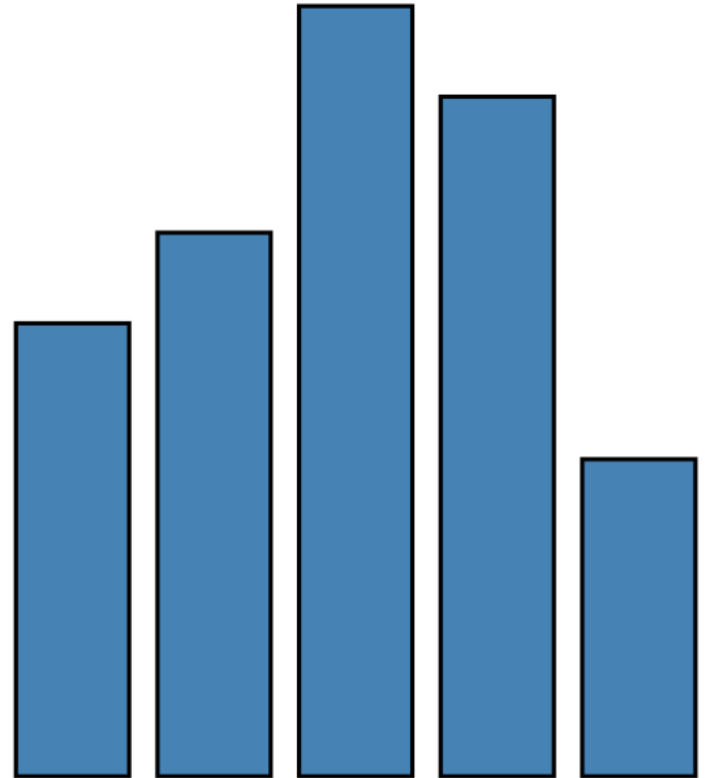
RECT

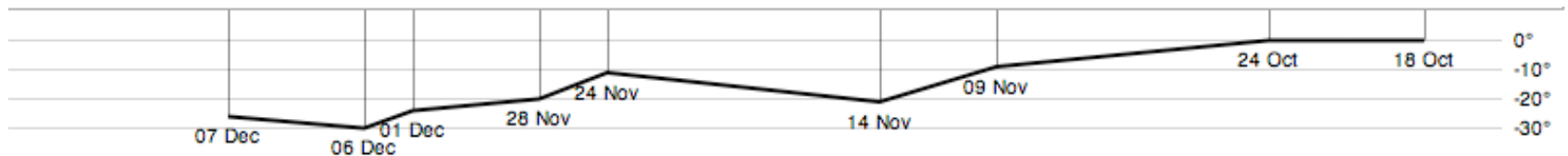
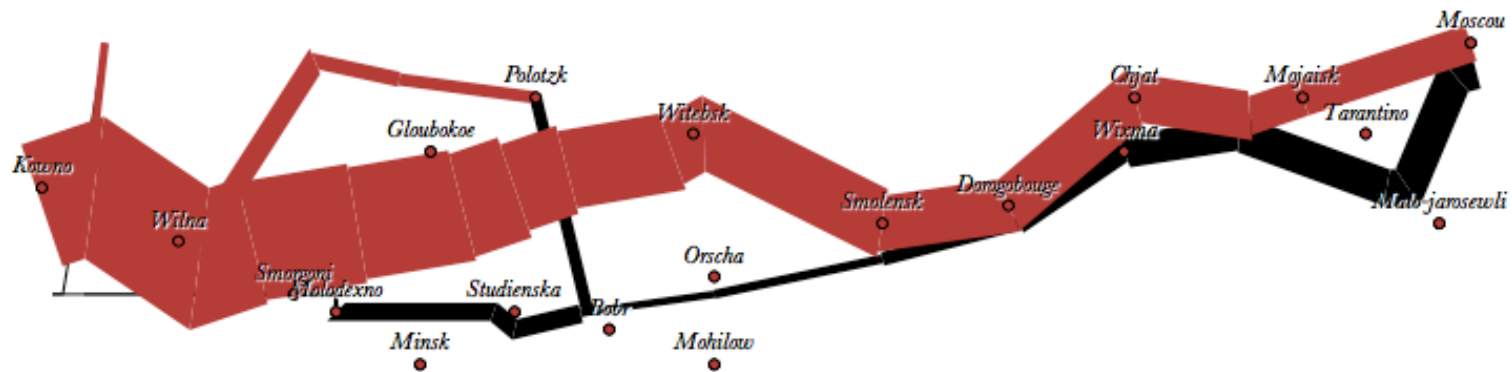
$\lambda : D \rightarrow R$

data	1	1.2	1.7	1.5	0.7
visible	true				
left	$\lambda: \text{index} * 25$				
bottom	0				
width	20				
height	$\lambda: \text{datum} * 80$				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



```
var vis = new pv.Panel();  
vis.add(pv.Bar)  
  .data([1, 1.2, 1.7, 1.5, 0.7])  
  .visible(true)  
  .left((d) => this.index * 25);  
  .bottom(0)  
  .width(20)  
  .height((d) => d * 80)  
  .fillStyle("blue")  
  .strokeStyle("black")  
  .lineWidth(1.5);  
vis.render();
```





```
var army = pv.nest(napoleon.army, "dir", "group");
var vis = new pv.Panel();
```

```
var lines = vis.add(pv.Panel).data(army);
lines.add(pv.Line)
  .data(() => army[this.idx])
  .left(lon).top(lat).size((d) => d.size/8000)
  .strokeStyle(() => color[army[panelIndex][0].dir]);
```

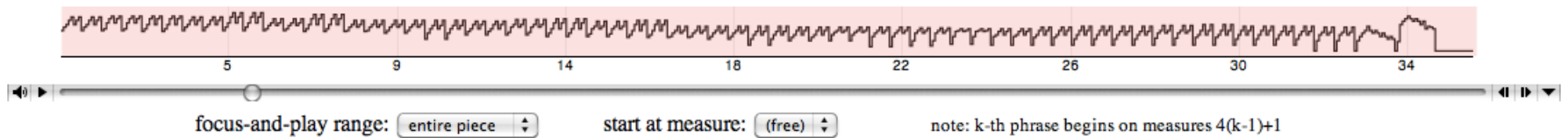
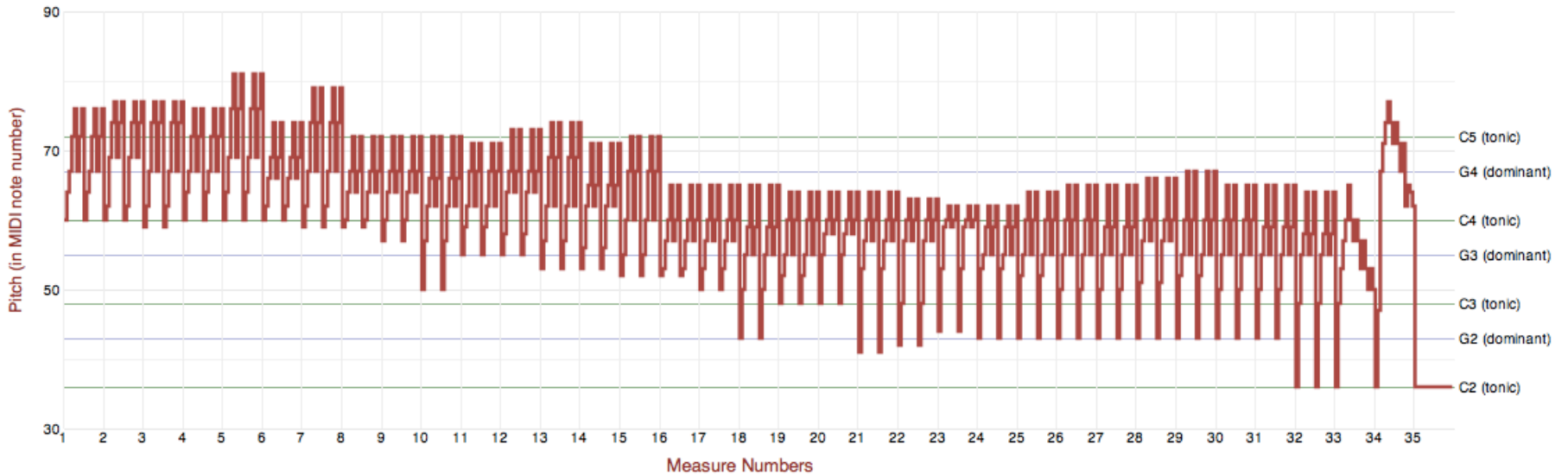
```
vis.add(pv.Label).data(napoleon.cities)
  .left(lon).top(lat)
  .text((d) => d.city).font("italic 10px Georgia")
  .textAlign("center").textBaseline("middle");
```

```
vis.add(pv.Rule).data([0,-10,-20,-30])
  .top((d) => 300 - 2*d - 0.5).left(200).right(150)
  .lineWidth(1).strokeStyle("#ccc")
  .anchor("right").add(pv.Label)
  .font("italic 10px Georgia")
  .text((d) => d+"°").textBaseline("center");

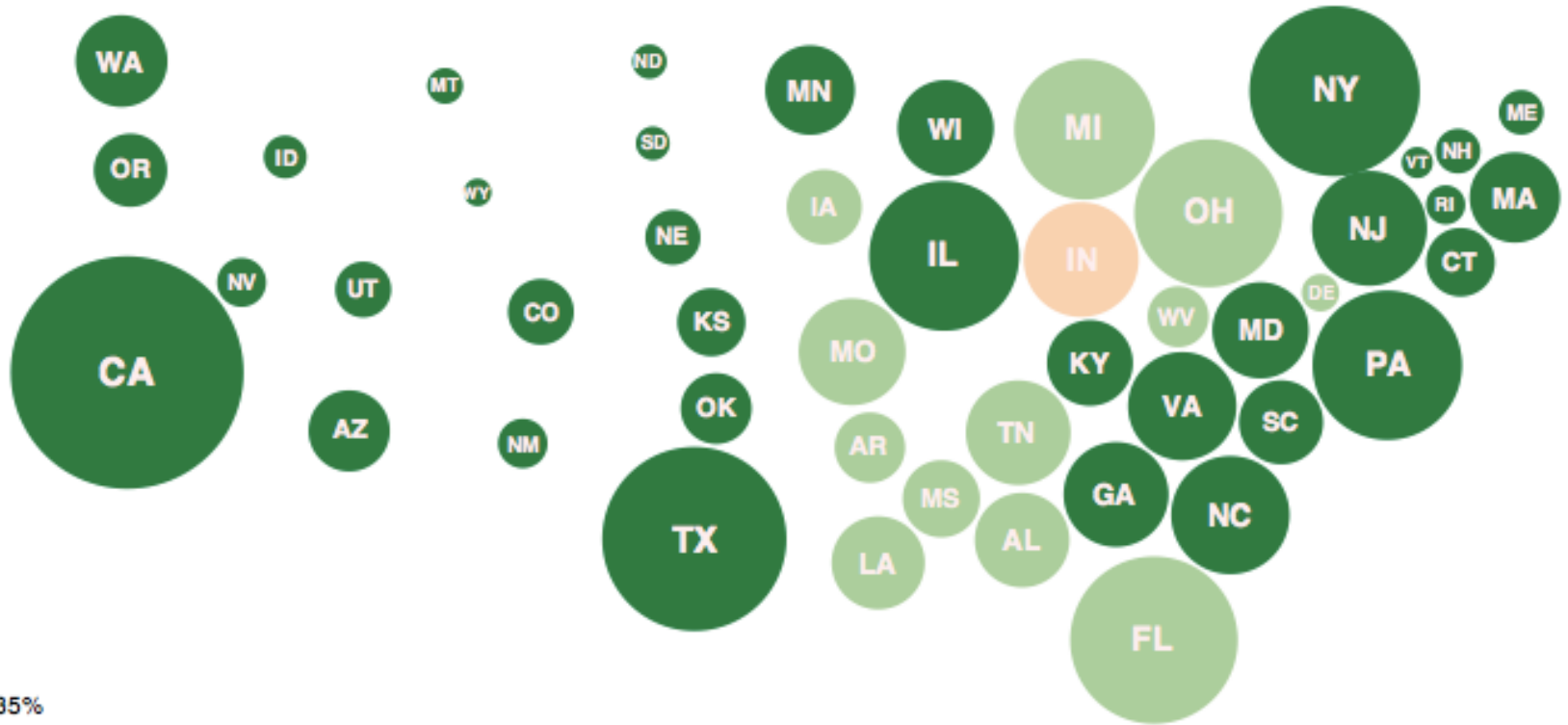
vis.add(pv.Line).data(napoleon.temp)
  .left(lon).top(tmp) .strokeStyle("#0")
  .add(pv.Label)
  .top((d) => 5 + tmp(d))
  .text((d) => d.temp+"° "+d.date.substr(0,6))
  .textBaseline("top").font("italic 10px Georgia");
```

**PRELUDE NO.1 IN C MAJOR, BWV 846
(FROM WELL-TEMPERED CLAVIER, BOOK 1)**

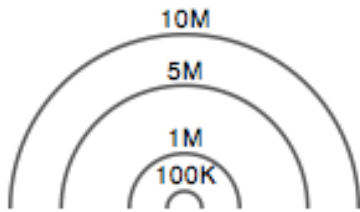
BY J.S. BACH

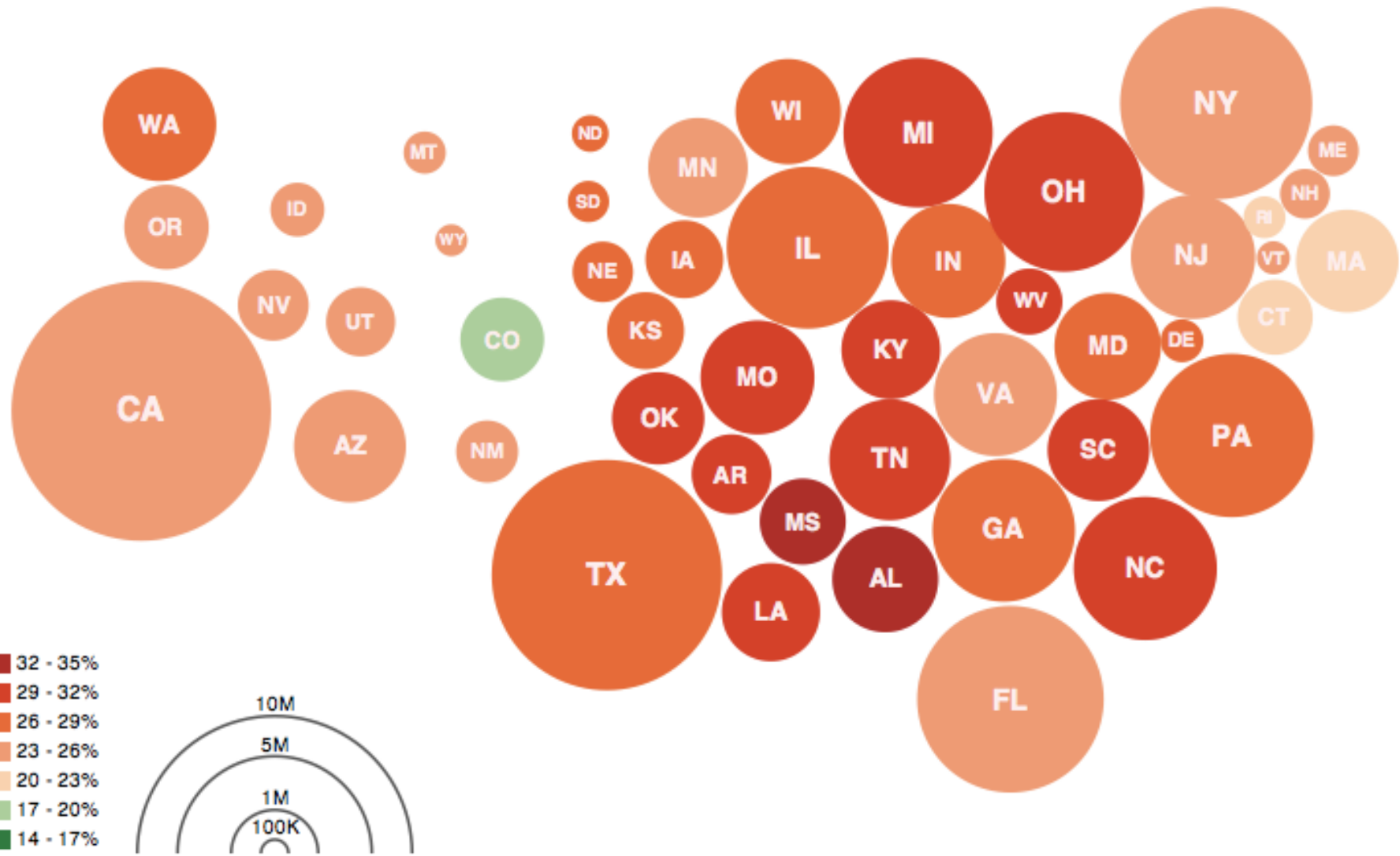


Bach's Prelude #1 in C Major | Jieun Oh

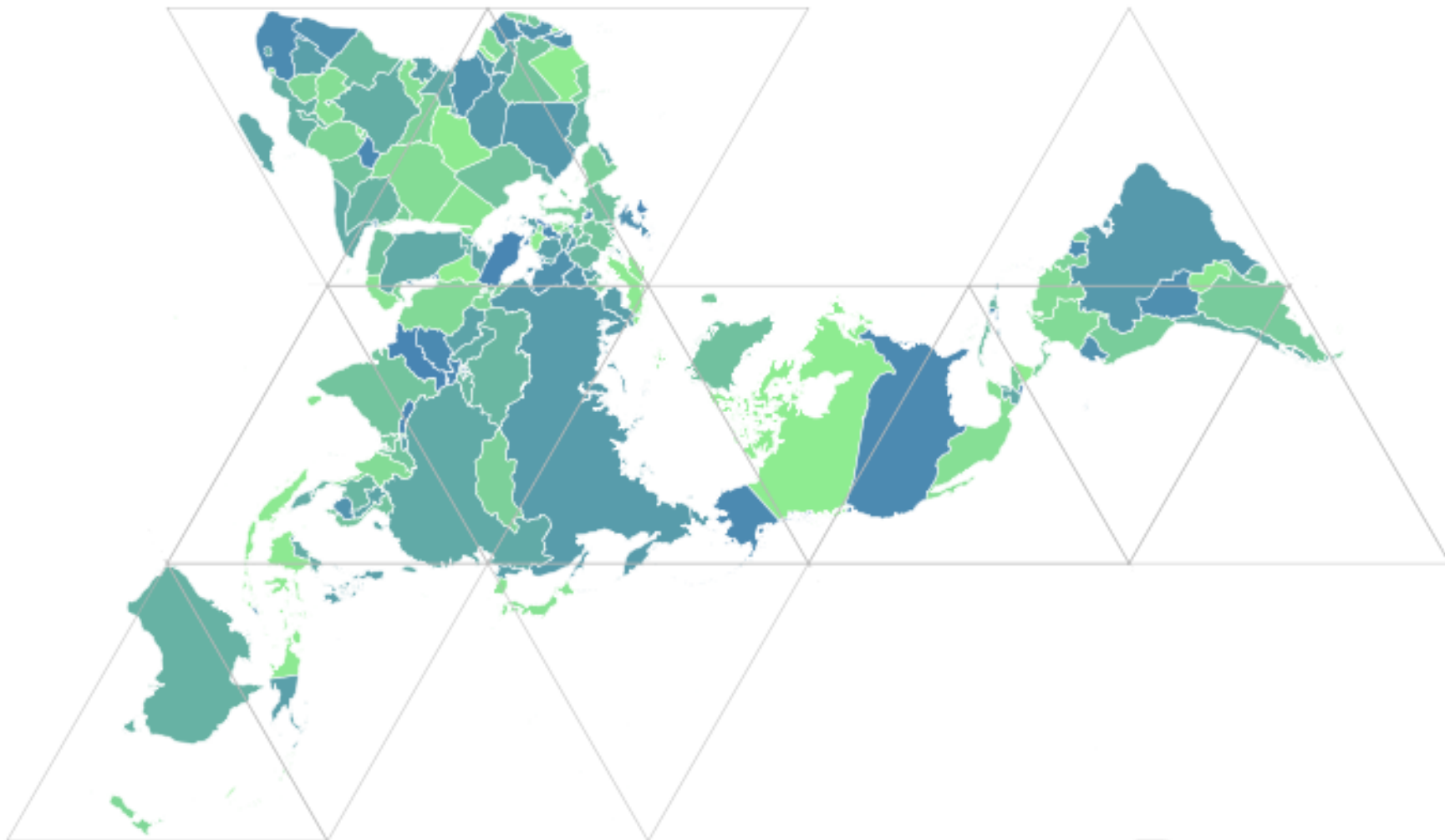


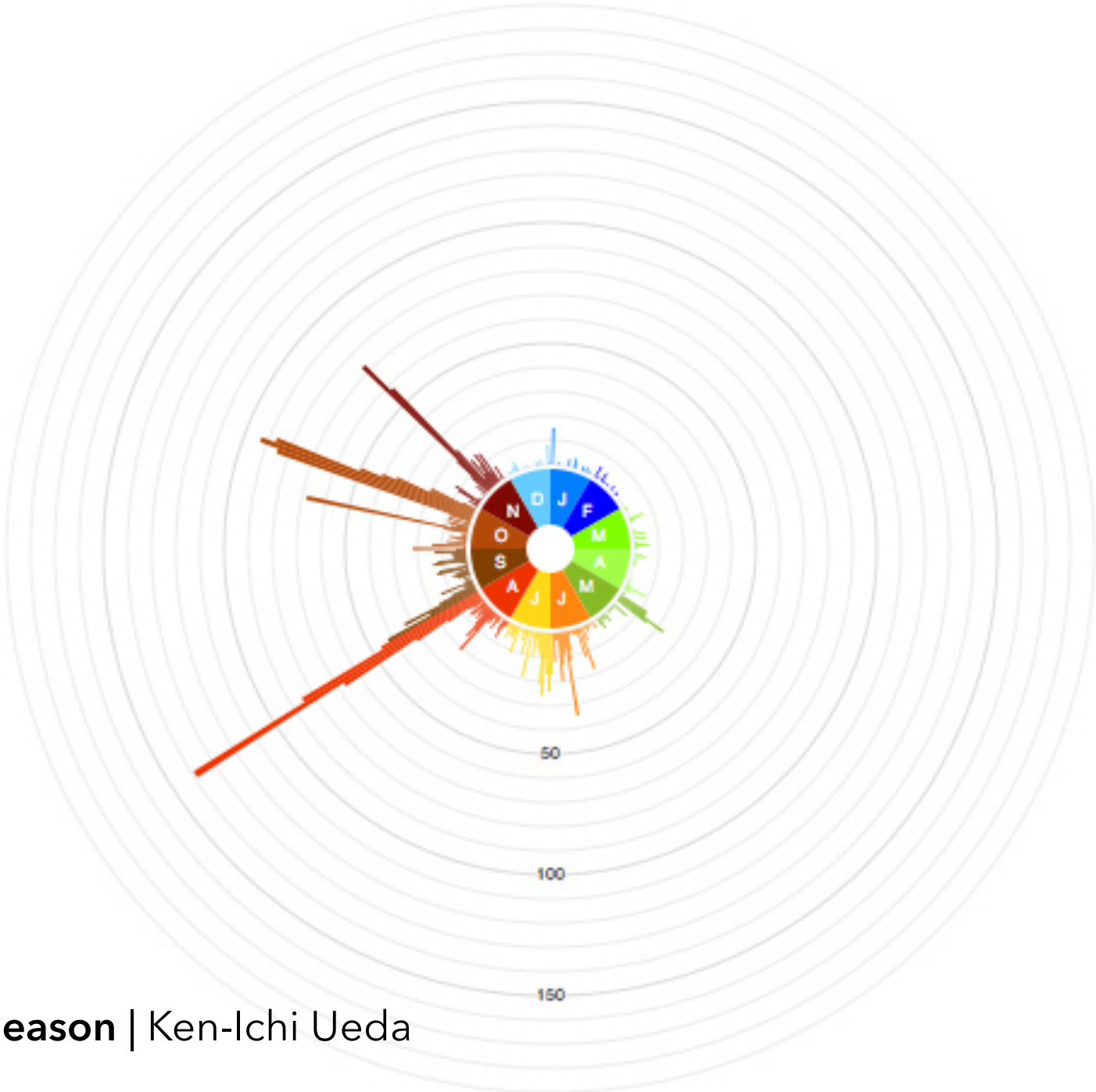
- 32 - 35%
- 29 - 32%
- 26 - 29%
- 23 - 26%
- 20 - 23%
- 17 - 20%
- 14 - 17%



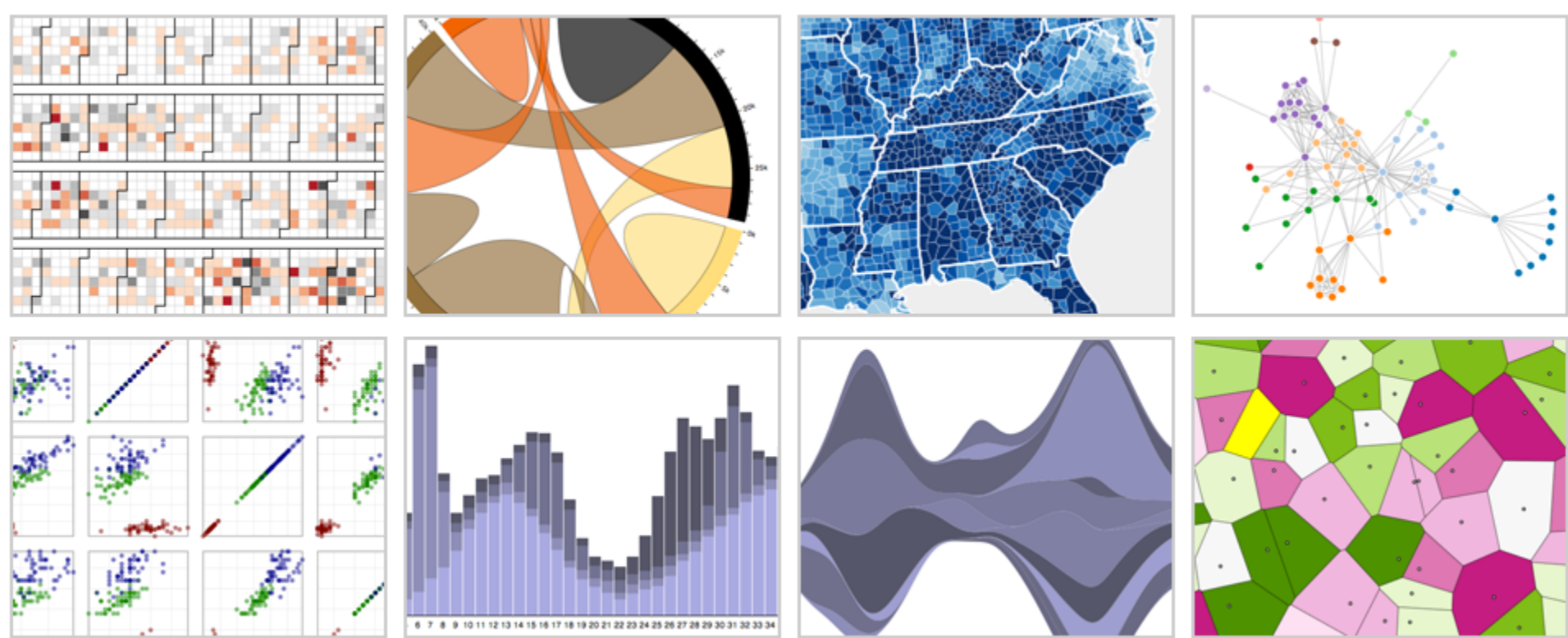


Obesity Map | Vadim Ogievetsky





d3.js Data-Driven Documents



with **Mike Bostock** & Vadim Ogievetsky

Protovis

Specialized mark types

- + Streamlined design
- Limits expressiveness
- More overhead (slower)
- Harder to debug
- Self-contained model

Specify a scene (nouns)

- + Quick for static vis
- Delayed evaluation
- Animation, interaction
are more cumbersome

Protovis

Specialized mark types

- + Streamlined design
- Limits expressiveness
- More overhead (slower)
- Harder to debug
- Self-contained model

Specify a scene (nouns)

- + Quick for static vis
- Delayed evaluation
- Animation, interaction
are more cumbersome

D3

Bind data to DOM

- Exposes SVG/CSS/...
- + Exposes SVG/CSS/...
- + Less overhead (faster)
- + Debug in browser
- + Use with other tools

Transform a scene (verbs)

- More complex model
- + Immediate evaluation
- + Dynamic data, anim,
and interaction natural

D3 Selections

The core abstraction in D3 is a *selection*.

```
// Add and configure an SVG element
var 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 Binding

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

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

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

```
var 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.
```

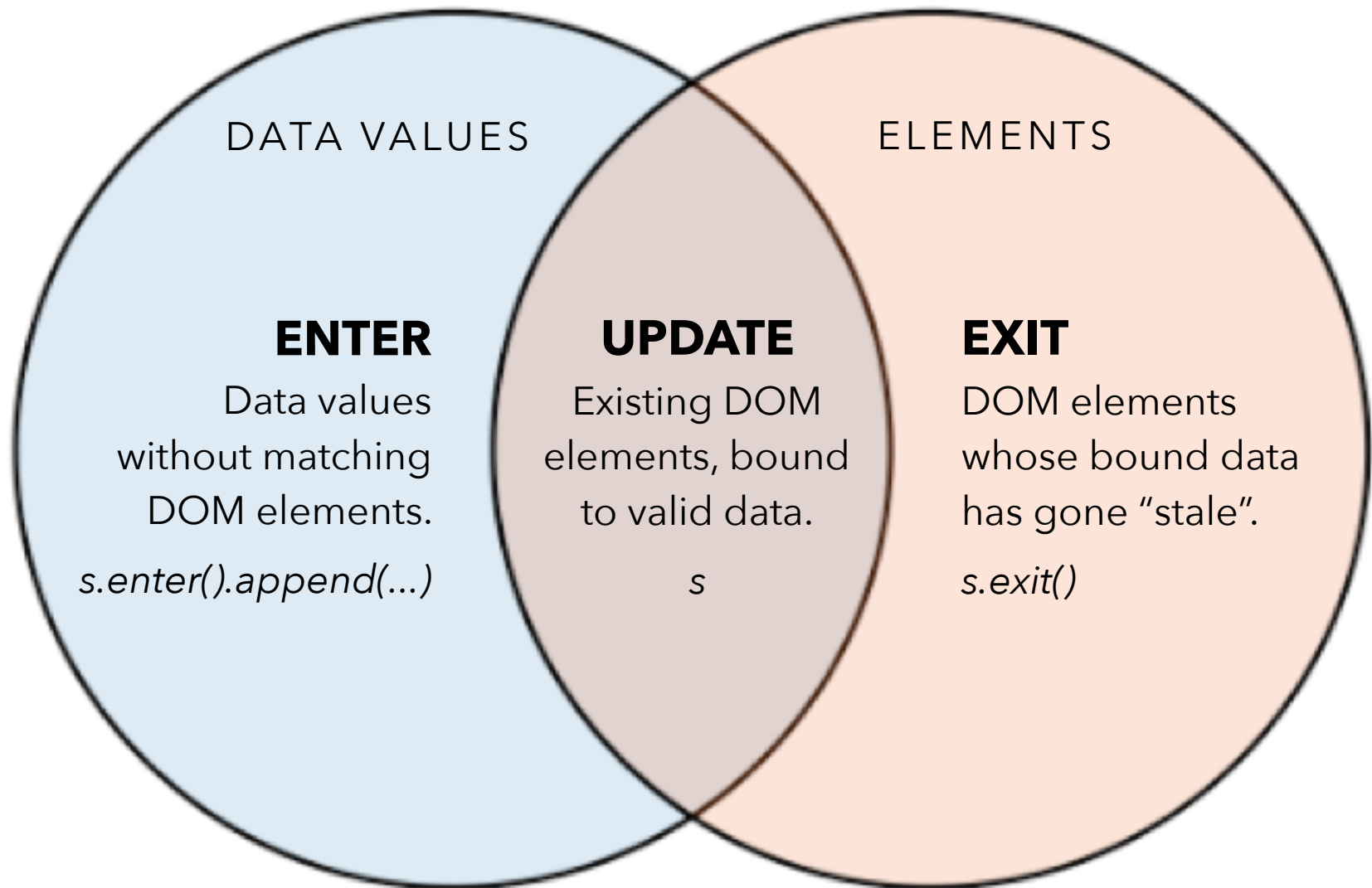
```
bars.enter().append("rect").attr("class", "bars");
```

```
// 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();
```

The Data Join

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



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!

Ease-of-Use



Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2

Visualization Grammars

Protovis, D3.js

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness



Administrivia

A2: Exploratory Data Analysis

Use visualization software to form & answer questions

First steps:

Step 1: Pick domain & data

Step 2: Pose questions

Step 3: Profile the data

Iterate as needed

Create a dashboard

Interact with controls

Refine your questions

Author a report

Screenshots of most insightful views (10+)

Include titles and captions for each view



Due by 11:59pm

Monday, Apr 22

D3.js Tutorial

Date: **Wednesday, April 24**

Time: **3:30pm to 5:00pm**

Location: **Gates Center G20**

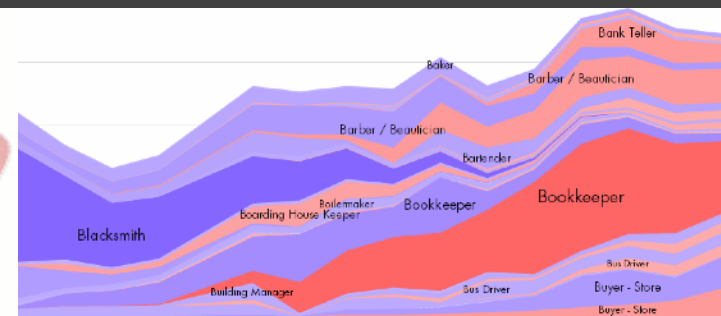
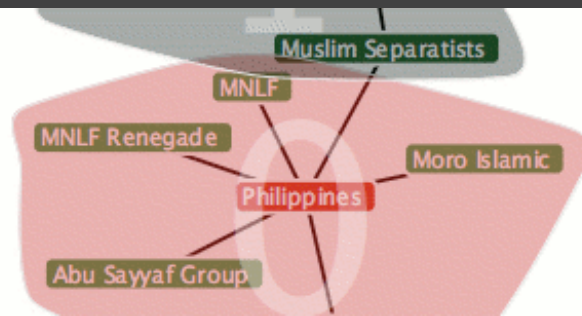
D3.js is a popular JavaScript visualization library, valuable for A3 and your Final Project...

A3: Interactive Prototype

Create an interactive visualization. Choose a driving question for a dataset and develop an appropriate visualization + interaction techniques, then deploy your visualization on the web.

Due by *11:59pm* on **Monday, May 6.**

Work in project teams of 2-4 people.

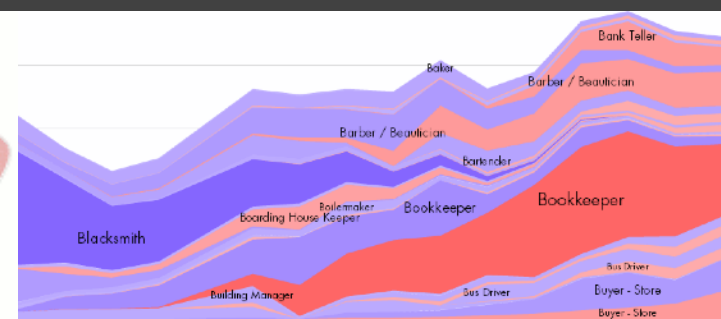
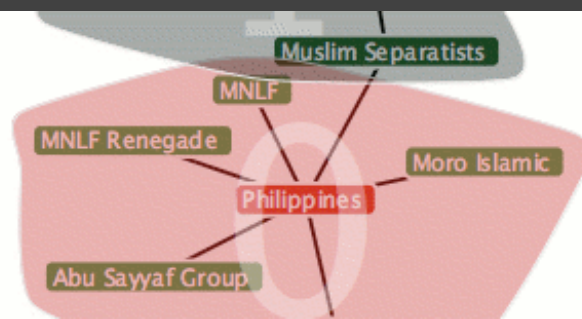


Requirements

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 is encouraged, but not required. Deploy your visualization using GitHub pages.

Write-up. Provide design rationale on your web page.



A3 Project Team

Form a **team of 2-4** for A3 *ASAP*.

(Start thinking about your Final Project, too!)

A3 is open-ended, but you can use it to start exploring FP topics if you like.

Submit signup form by **Friday 4/26, 11:59pm**.

If you do not have team mates, you should:

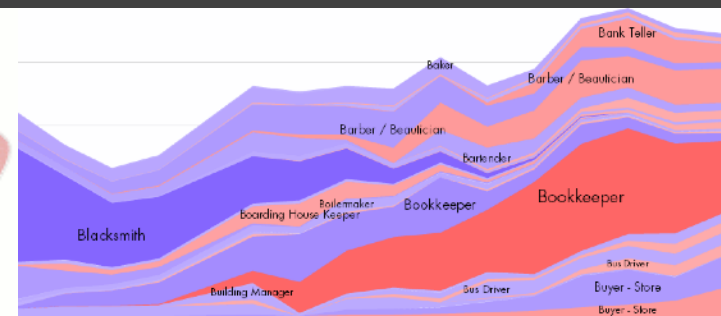
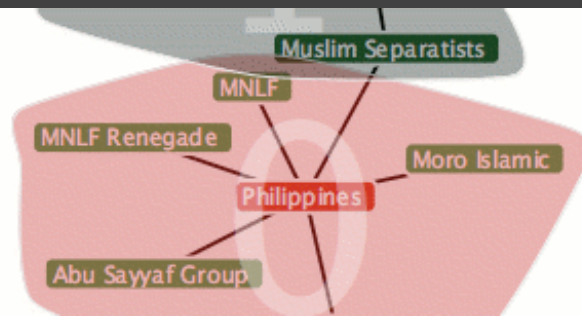
- Use the facilities on Canvas
- Stay after class/tutorial to meet potential partners

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

Visualization Grammars

Protovis, D3.js

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Chart Typologies

Excel, Many Eyes, Google Charts

Charting
Tools

Visual Analysis Grammars

VizQL, ggplot2

Declarative
Languages

Visualization Grammars

Protovis, D3.js

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Processing, OpenGL, Java2D

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)
  .enter().append("rect")
  .attr("x", function(d) { return xscale(d.foo); })
  .attr("y", function(d) { return yscale(d.bar); })
```

The New York Times

Tuesday, October 26, 2010 Last Update: 3:50 PM ET

Search [ING DIRECT](#)

— 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)

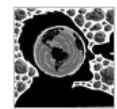


Joshua Kristal for The New York Times

Painting at 99, With No Compromises

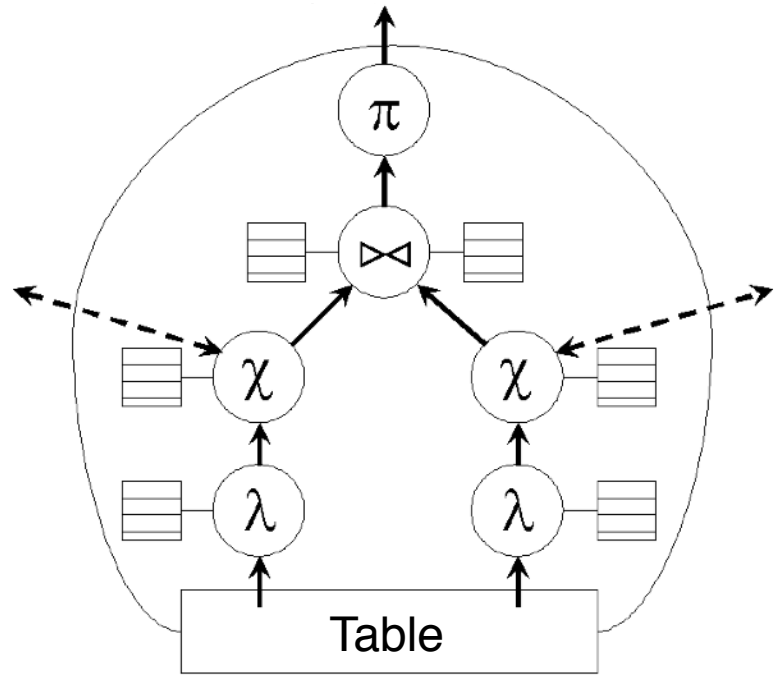
By ROBIN FINN
An exhibition celebrating Will Barnet's centennial year traces his evolution as a modern American artist.

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 » At 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"></scrip
    <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>
    </script>
    <span id="to_scrip"></span>
    <script type="text/javascript"></script>
    
    <script type="text/javascript" src="http://graphics8.nytimes.c
  
```

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?

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

Declarative
Languages

Visualization Grammars

Protovis, D3.js

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*

Declarative
Languages

Visualization Grammars

Protovis, 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*

Declarative
Languages

Visualization Grammars

Protovis, D3.js, *Vega*

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Processing, OpenGL, Java2D

Visual Analysis Grammars

VizQL, ggplot2, *Vega-Lite*

Declarative
Languages

Visualization Grammars

Protovis, 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

Protovis, D3.js, *Vega*

Component Architectures

Prefuse, Flare, Improvise, VTK

Programming
Toolkits

Graphics APIs

Processing, OpenGL, Java2D

VEGA-LITE

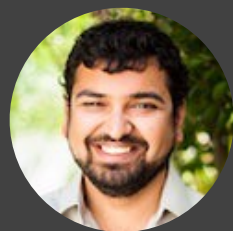
A Grammar of Interactive Graphics

Kanit "Ham" Wongsuphasawat @kanitw

Dominik Moritz @domoritz

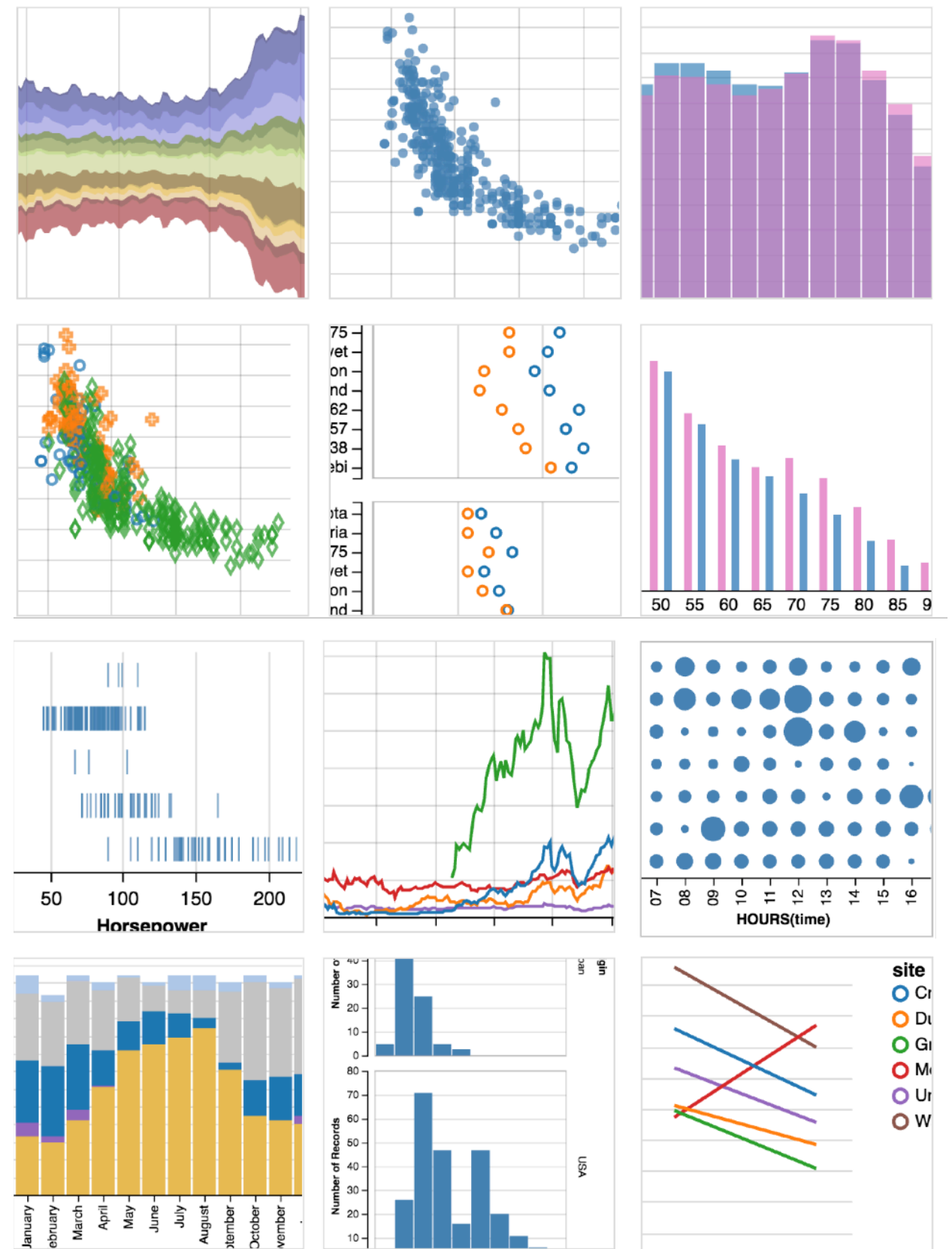
Arvind Satyanarayan @arvindsatya1

Jeffrey Heer @jeffrey_heer



Interactive Data Lab @uwdata

University of Washington



prefuse

Protovis

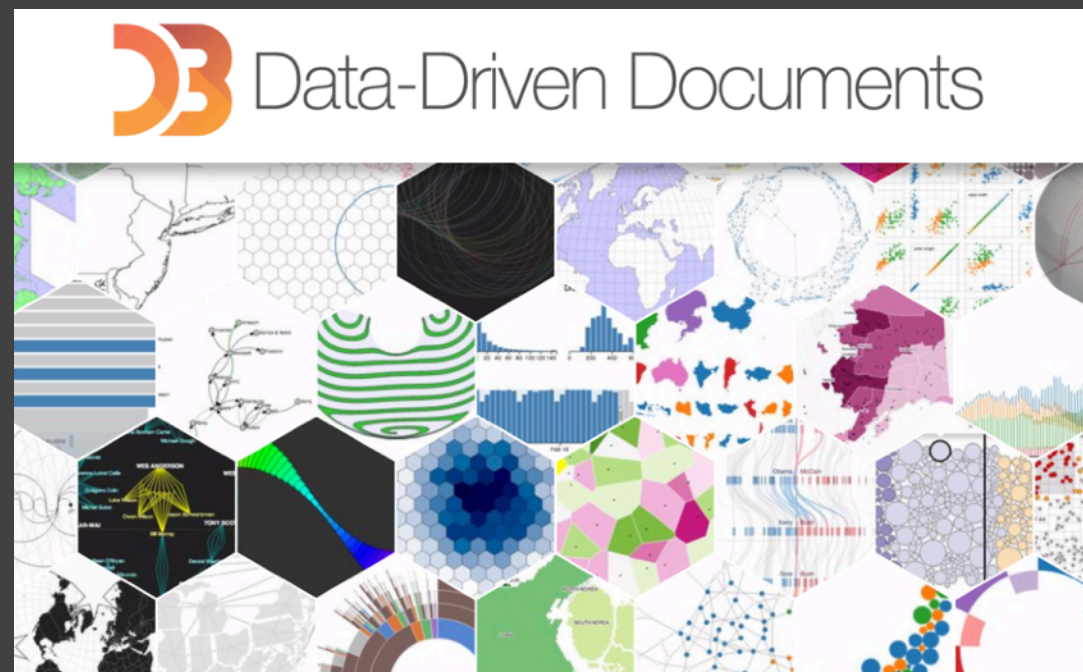


vega

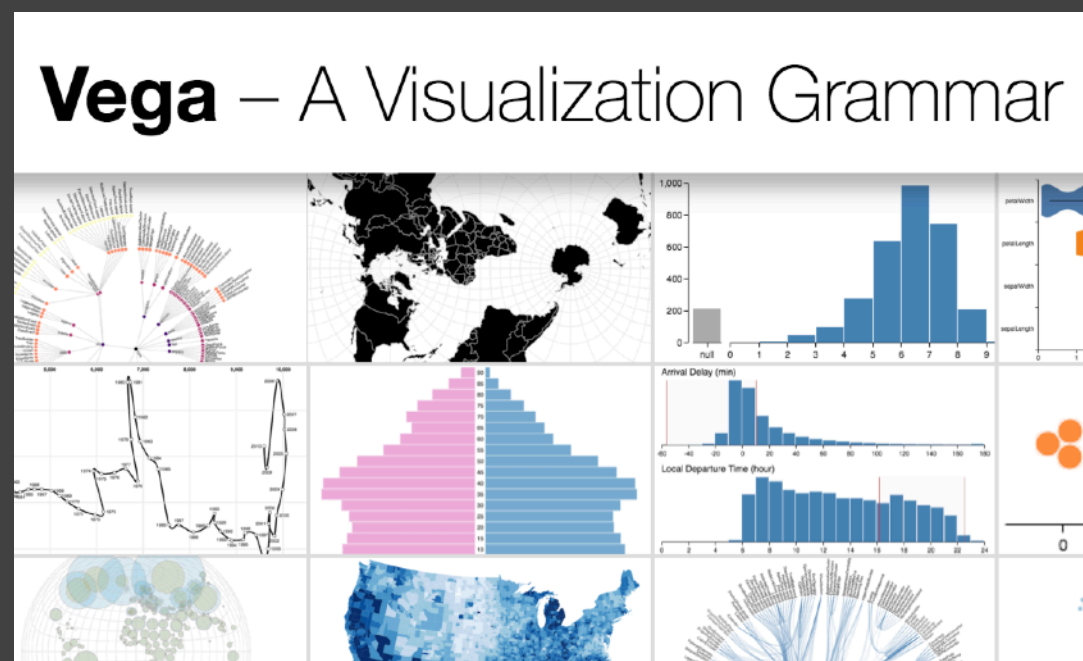


Data-Driven Documents

Grammar of Graphics *for Customized Designs*

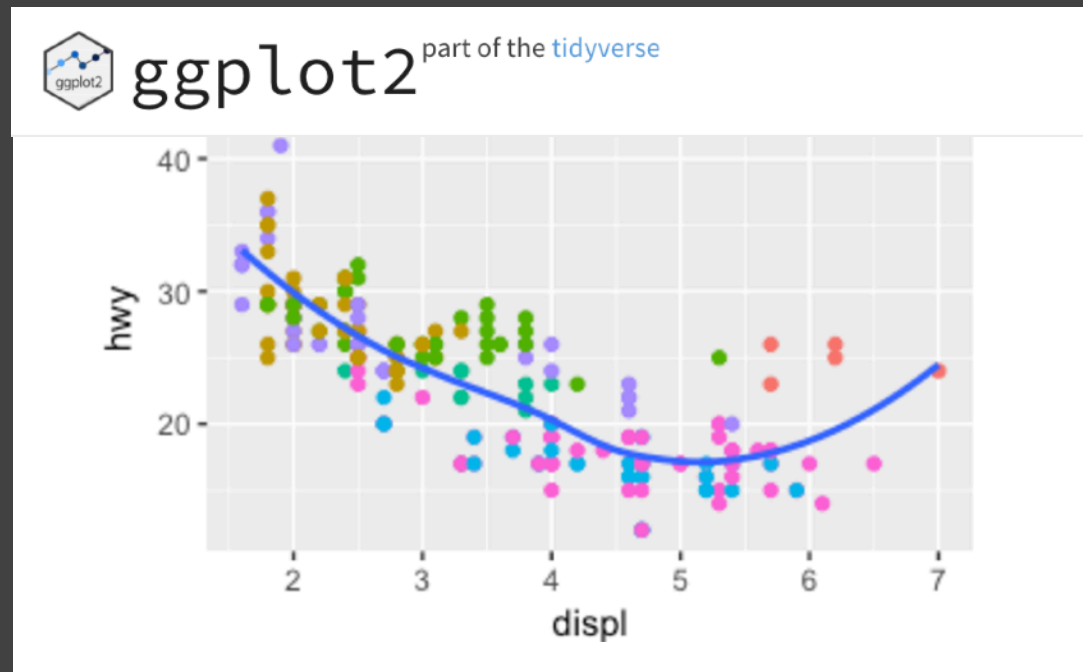


Offer **fine-grained control** for composing interactive graphics.



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

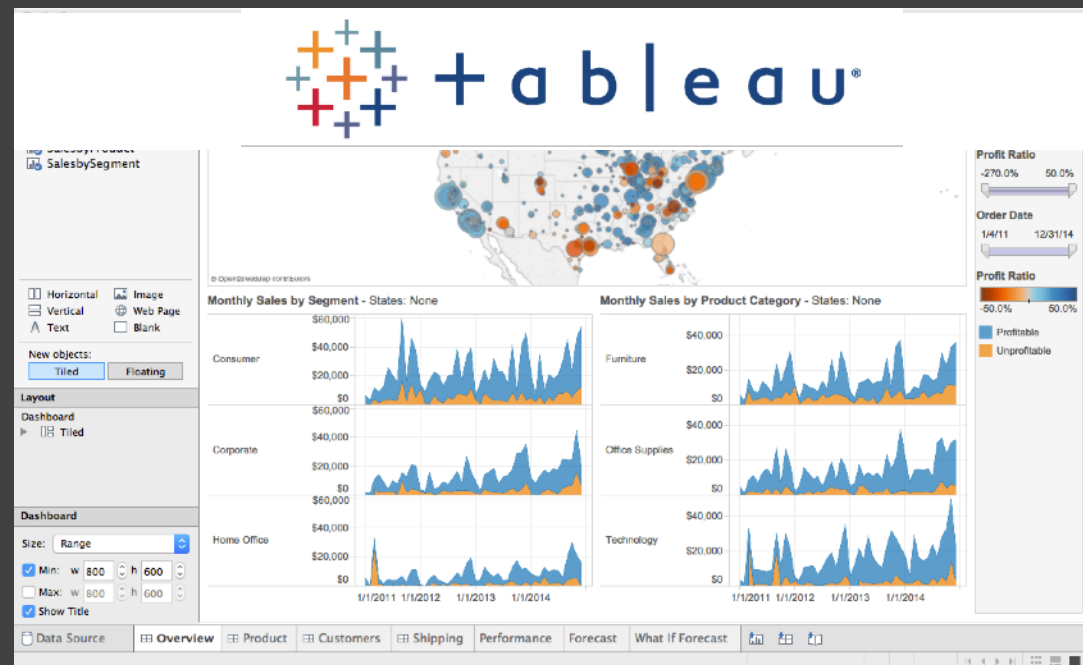
Grammar of Graphics *for Exploration*



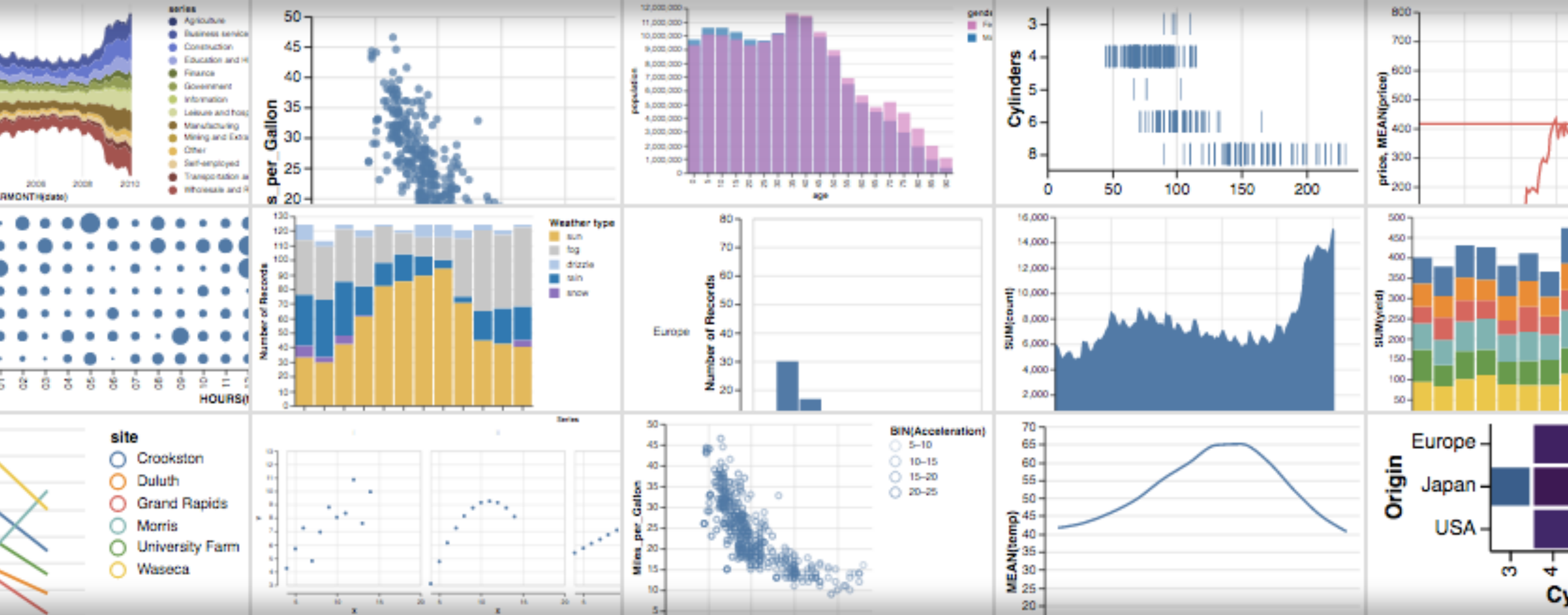
Facilitate **rapid exploration** with **concise** specifications by omitting low-level details.

Infer **sensible defaults** and allow customization by overriding defaults.

But **limited** support for **interactions**.



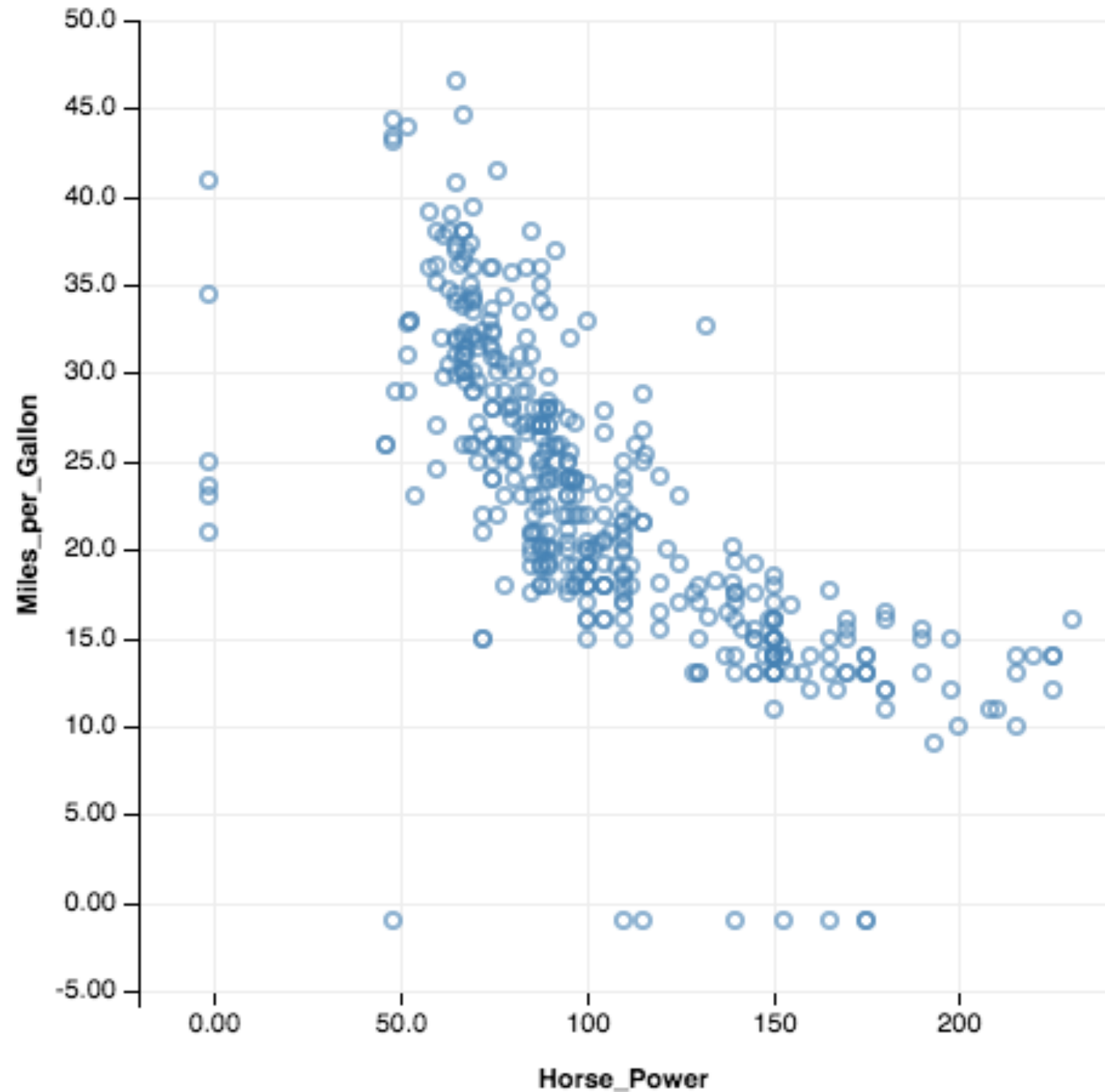
How might we author
interactive graphics in the
midst of analysis?



Vega-Lite: A Grammar of Interactive Graphics

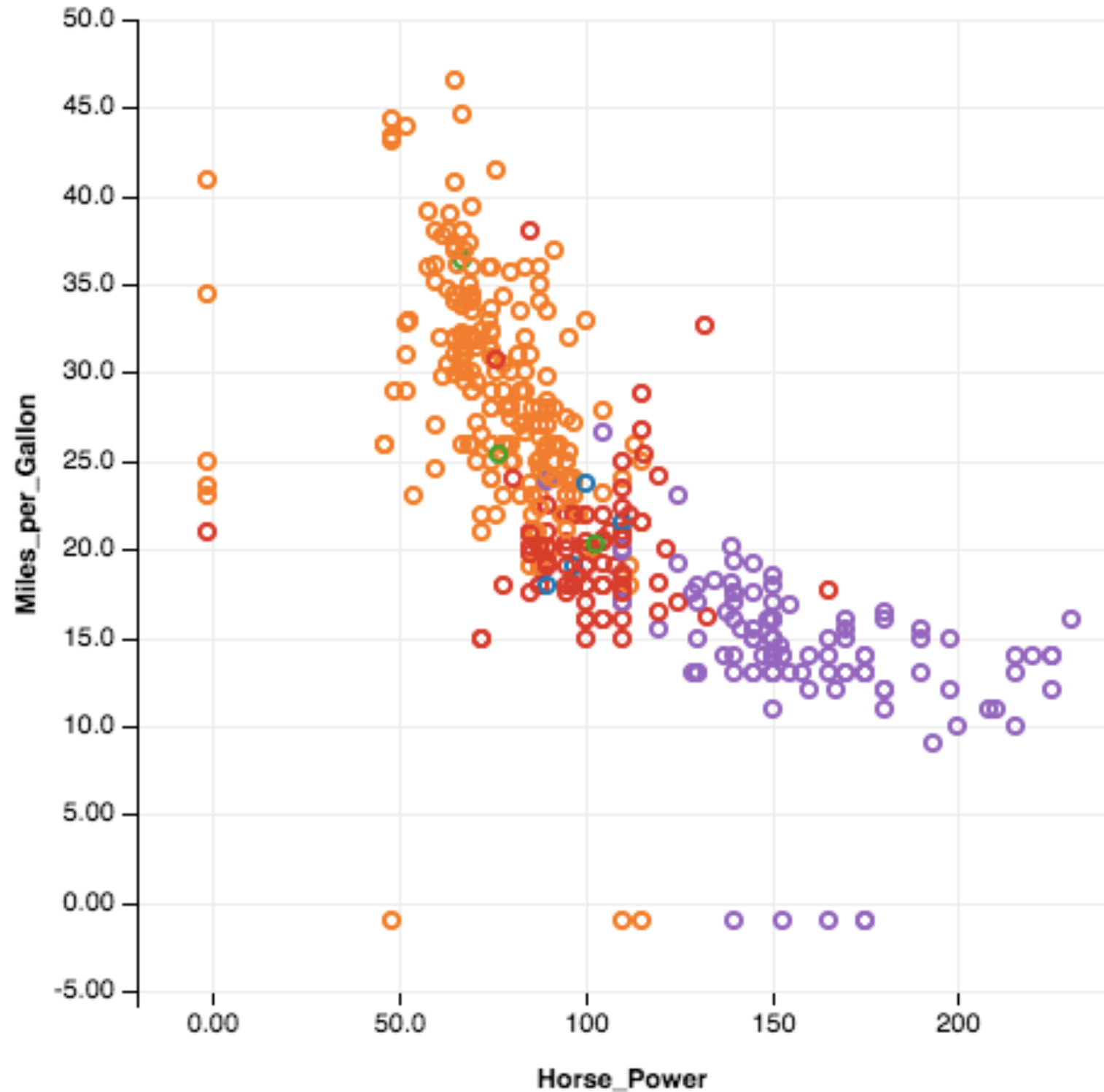
A. Satyanarayan, D. Moritz, K. Wongsuphasawat & J. Heer. *TVCG 2017*

Vega-Lite: Scatter Plot



```
{  
  "data": {"url": "data/cars.json"},  
  "mark": "point",  
  "encoding": {  
    "x": {"field": "Horse_Power", "type": "Q"},  
    "y": {"field": "Miles_per_Gallon", "type": "Q"}  
  }  
}
```

Vega-Lite: Scatter Plot

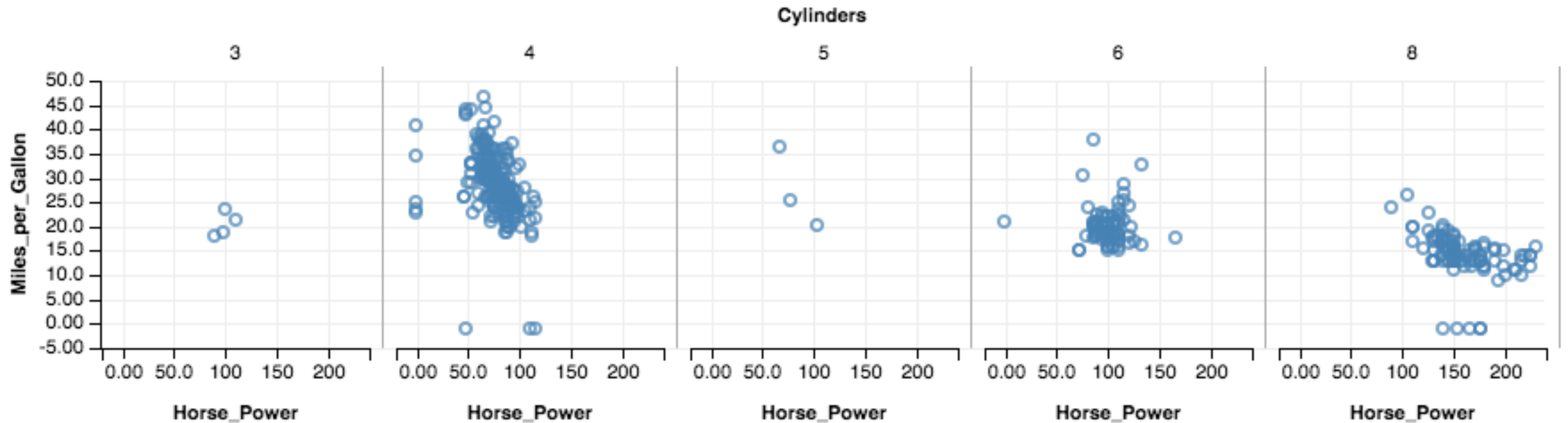


```
{  
  "data": {"url": "data/cars.json"},  
  "mark": "point",  
  "encoding": {  
    "x": {"field": "Horse_Power", "type": "Q"},  
    "y": {"field": "Miles_per_Gallon", "type": "Q"},  
    "color": {"field": "Cylinders", "type": "N"}  
  }  
}
```

Cylinders

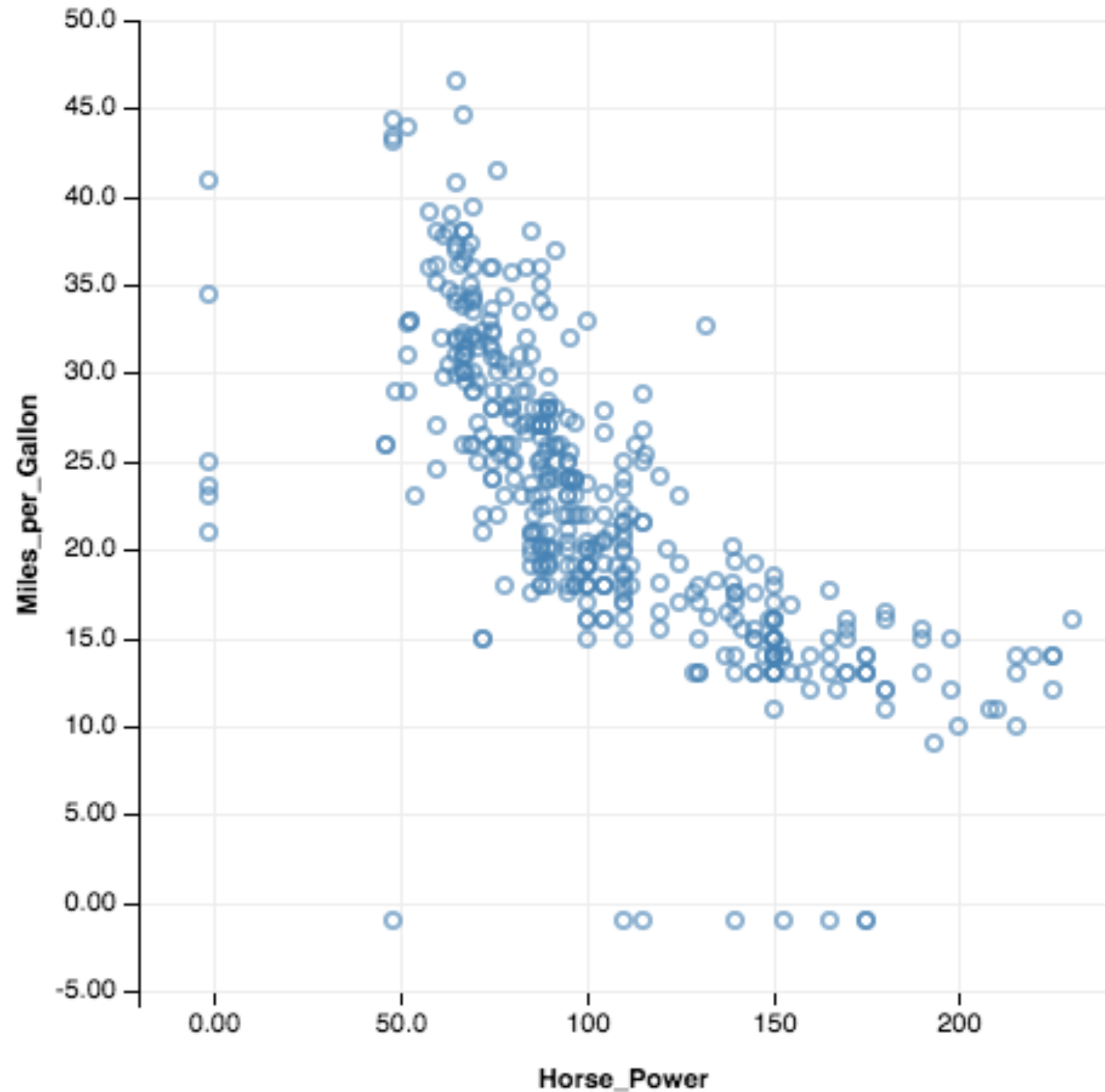
- 3
- 4
- 5
- 6
- 8

Vega-Lite: Trellis Plot



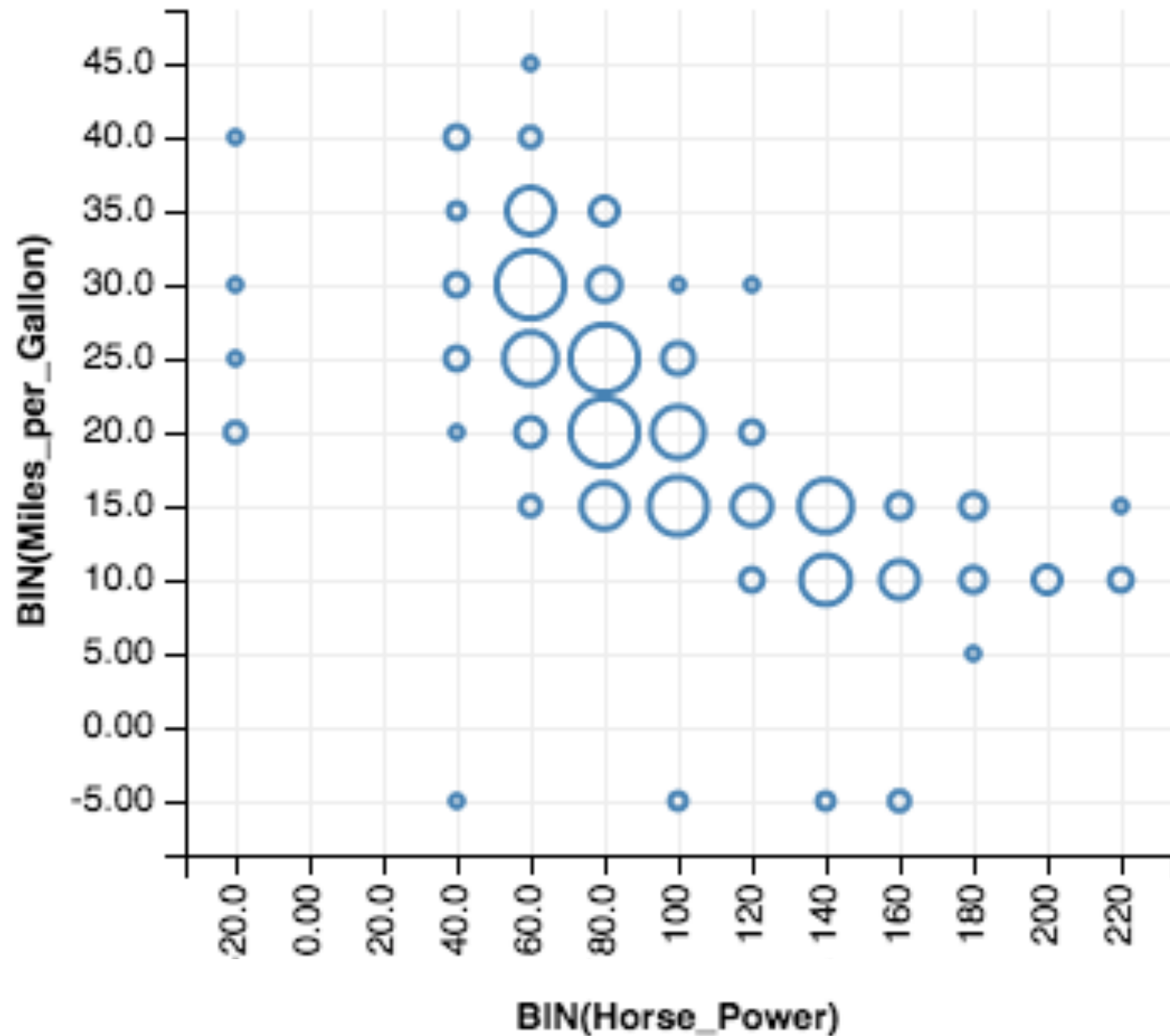
```
{  
  "data": {"url": "data/cars.json"},  
  "mark": "point",  
  "encoding": {  
    "x": {"field": "Horse_Power", "type": "Q"},  
    "y": {"field": "Miles_per_Gallon", "type": "Q"},  
    "column": {"field": "Cylinders", "type": "N"}  
  }  
}
```

Vega-Lite: Scatter Plot



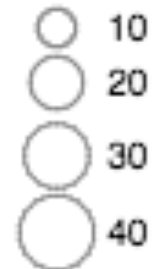
```
{  
  "data": {"url": "data/cars.json"},  
  "mark": "point",  
  "encoding": {  
    "x": {"field": "Horse_Power", "type": "Q"},  
    "y": {"field": "Miles_per_Gallon", "type": "Q"}  
  }  
}
```

Vega-Lite: 2D Histogram



```
{  
  "data": {"url": "data/cars.json"},  
  "mark": "point",  
  "encoding": {  
    "x": {"field": "Horse_Power", "type": "Q", "bin": true},  
    "y": {"field": "Miles_per_Gallon", "type": "Q", "bin": true},  
    "size": {"field": "*", "type": "Q", "aggregate": "count"}  
  }  
}
```

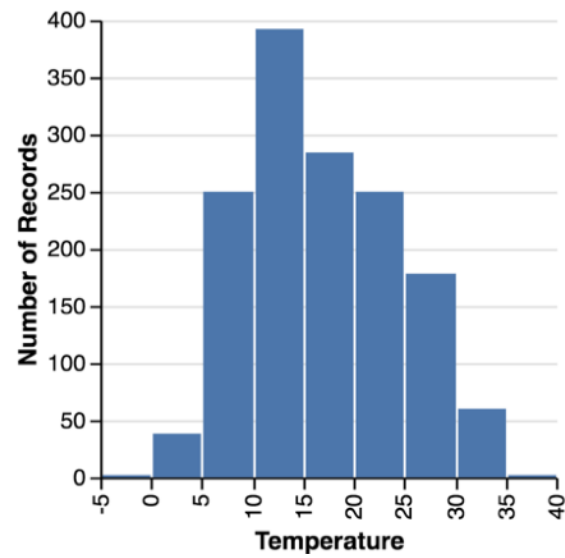
Number of Records



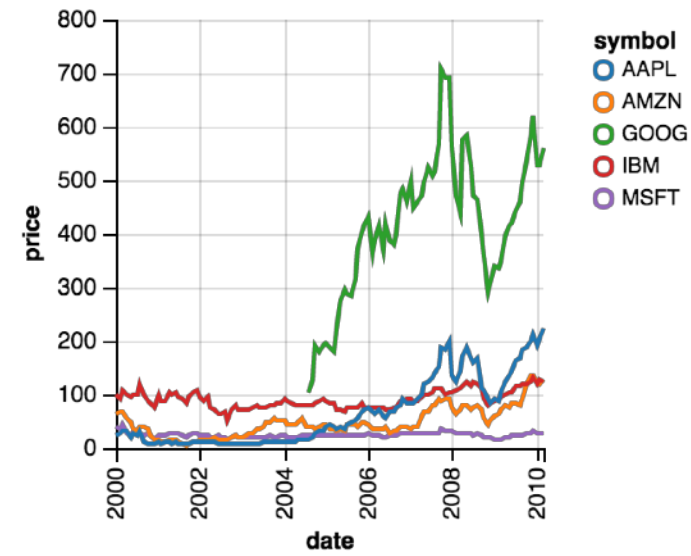
RESEARCH GOAL:

Extend grammars of statistical graphics to enable **multi-view composition** and **interaction**.

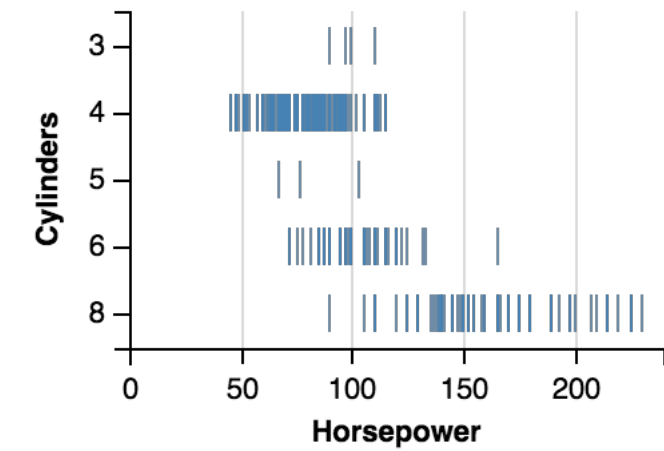
Histogram



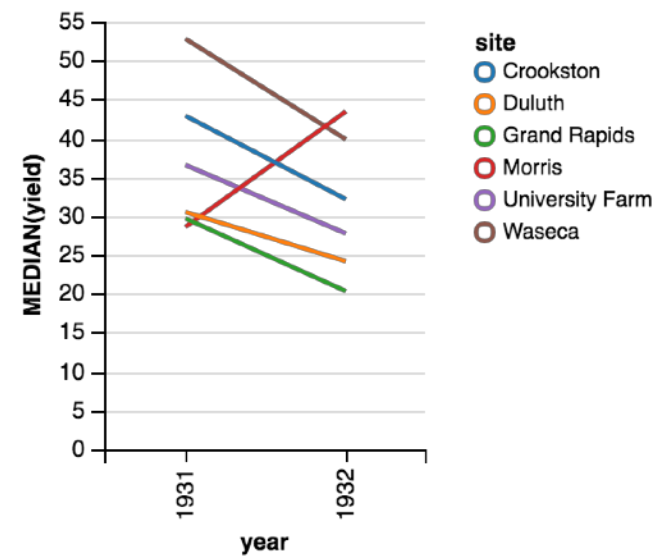
Line Chart



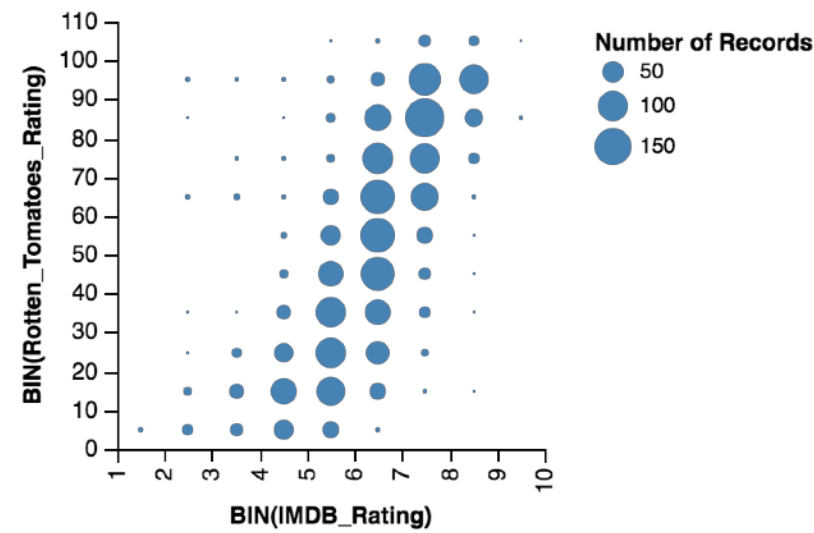
Strip Plot



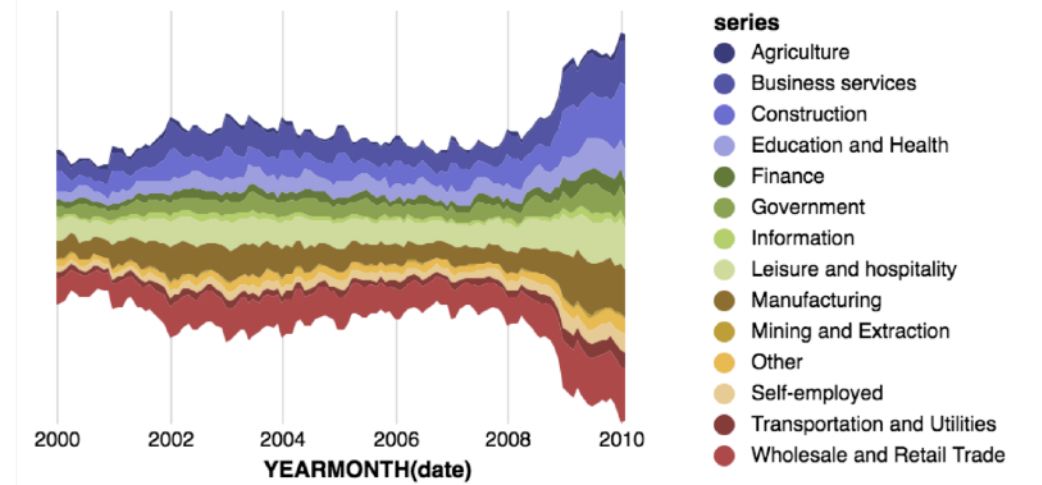
Slope Graph



Binned Scatter Plot

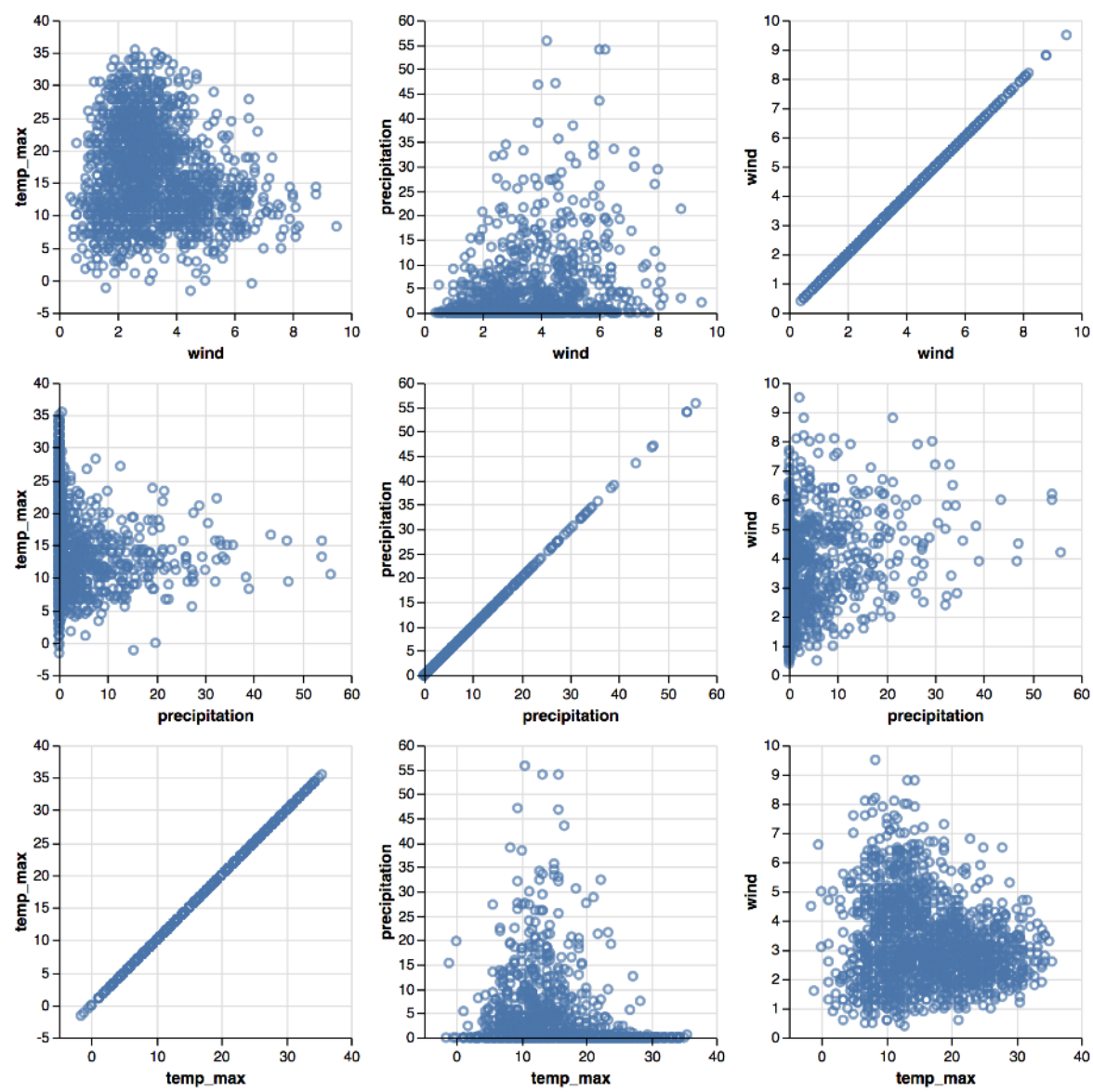


Area Chart

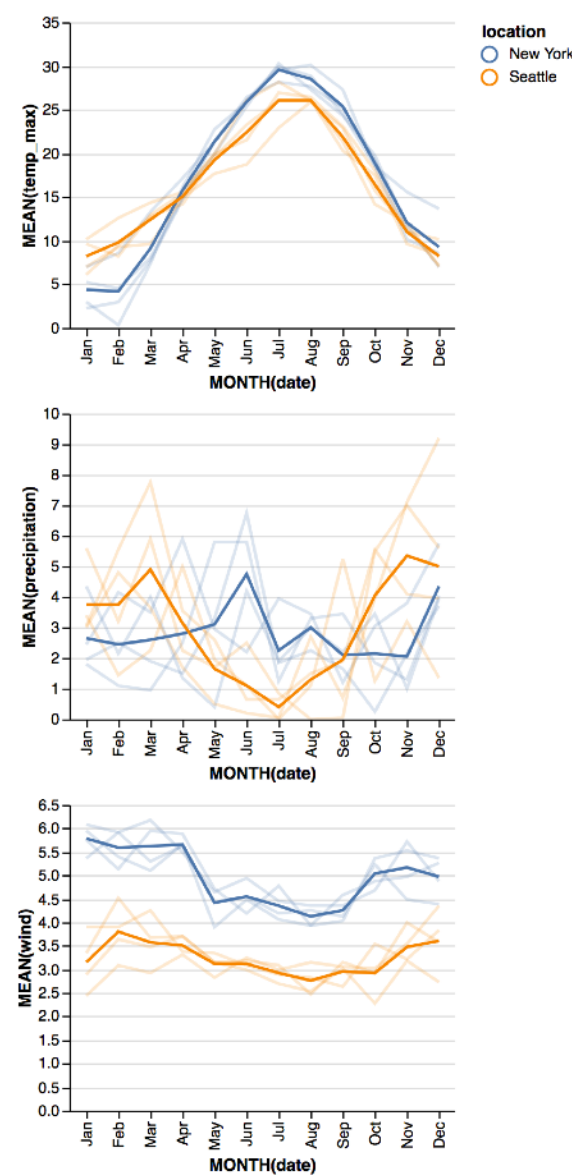


Vega-Lite: A Grammar of Graphics

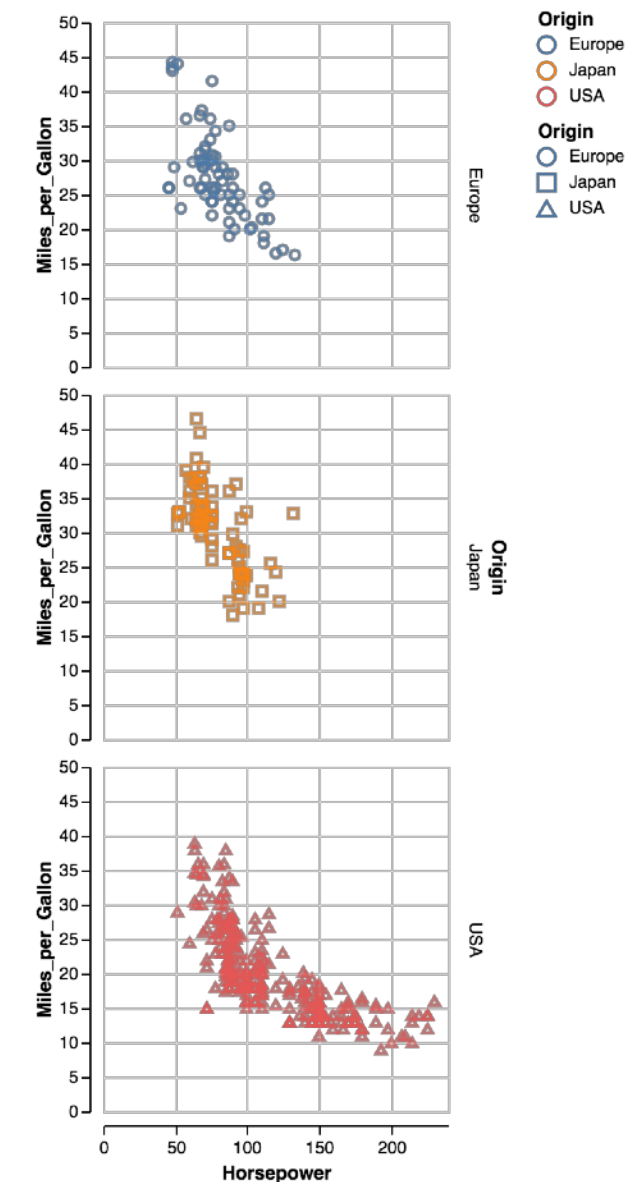
Scatter Plot Matrix



Concatenated & Layered View

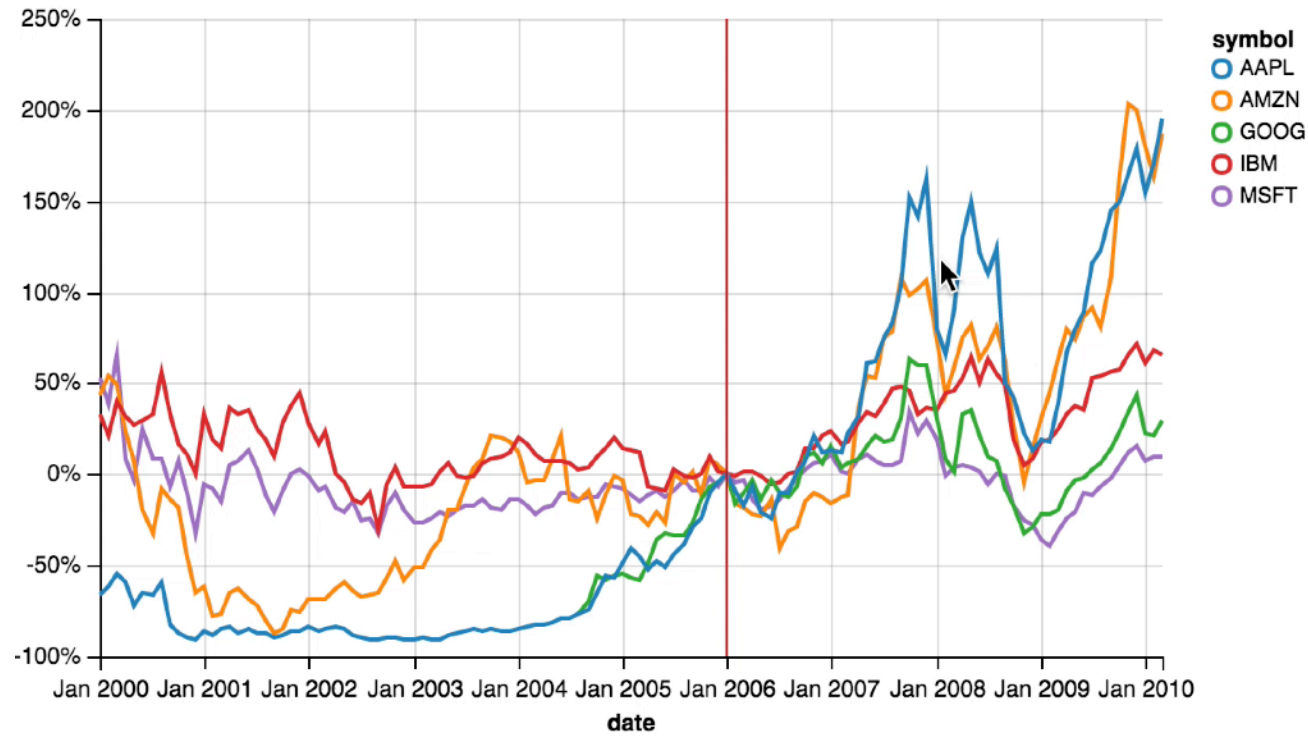


Faceted View

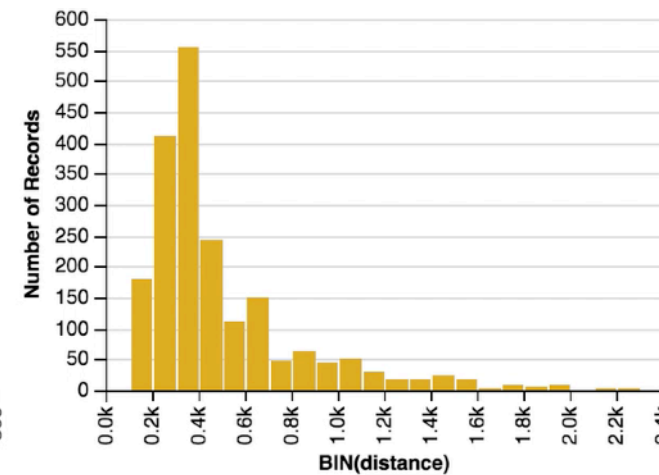
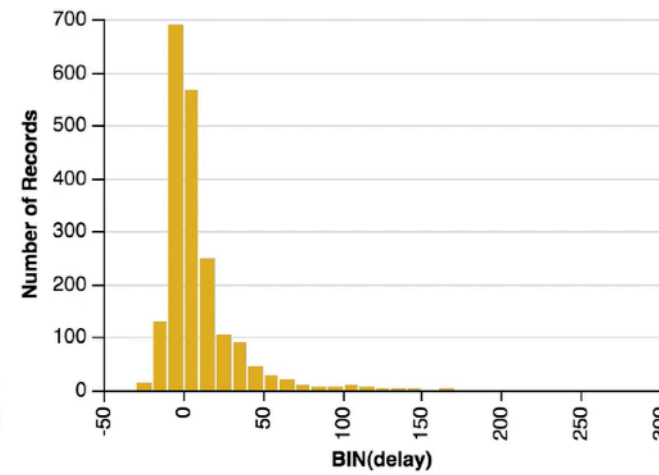
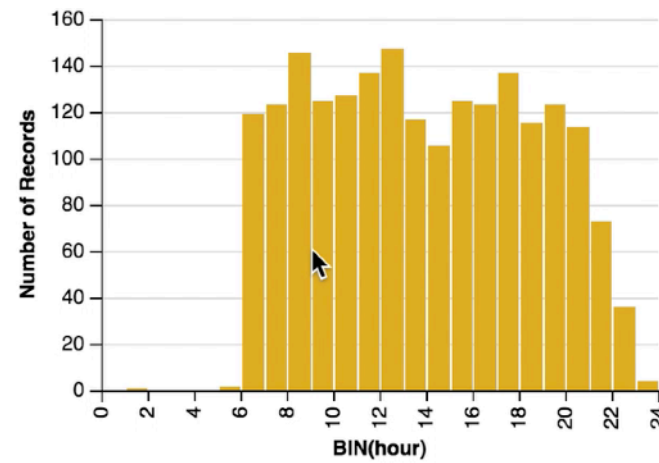
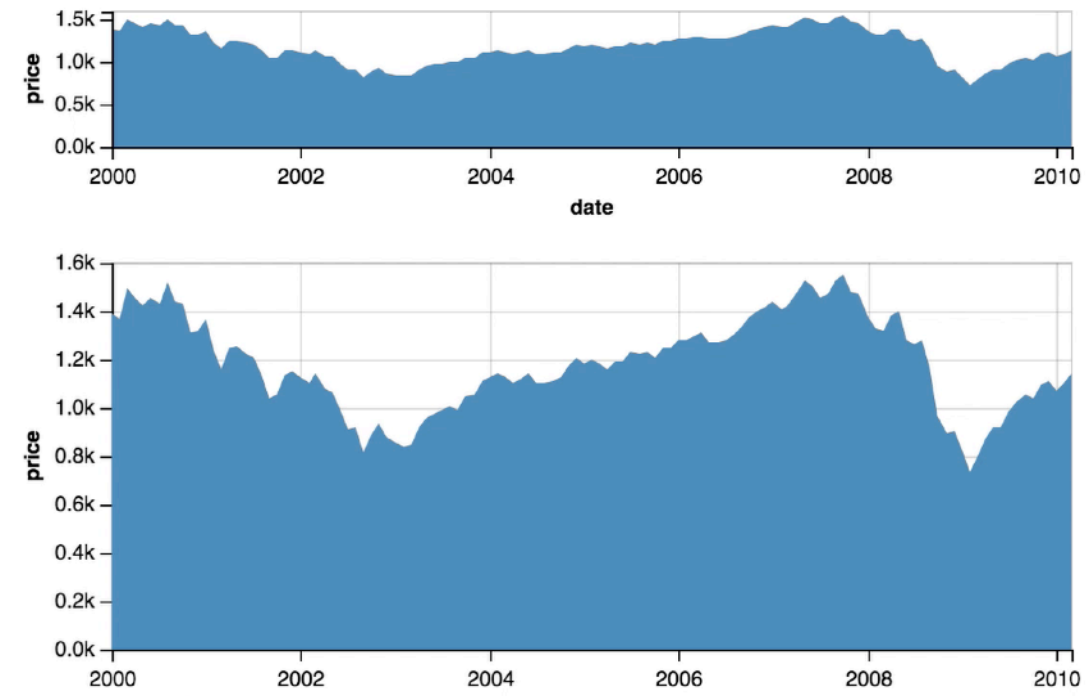


Vega-Lite: A Grammar of Multi-View Graphics

Indexed Chart

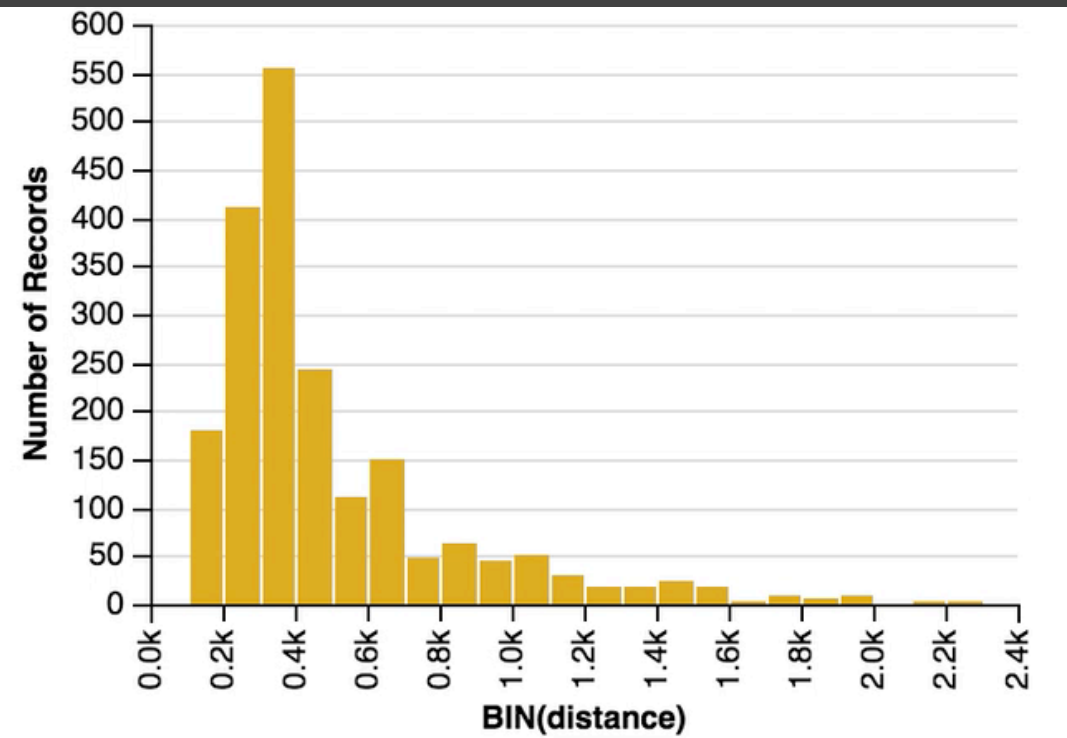
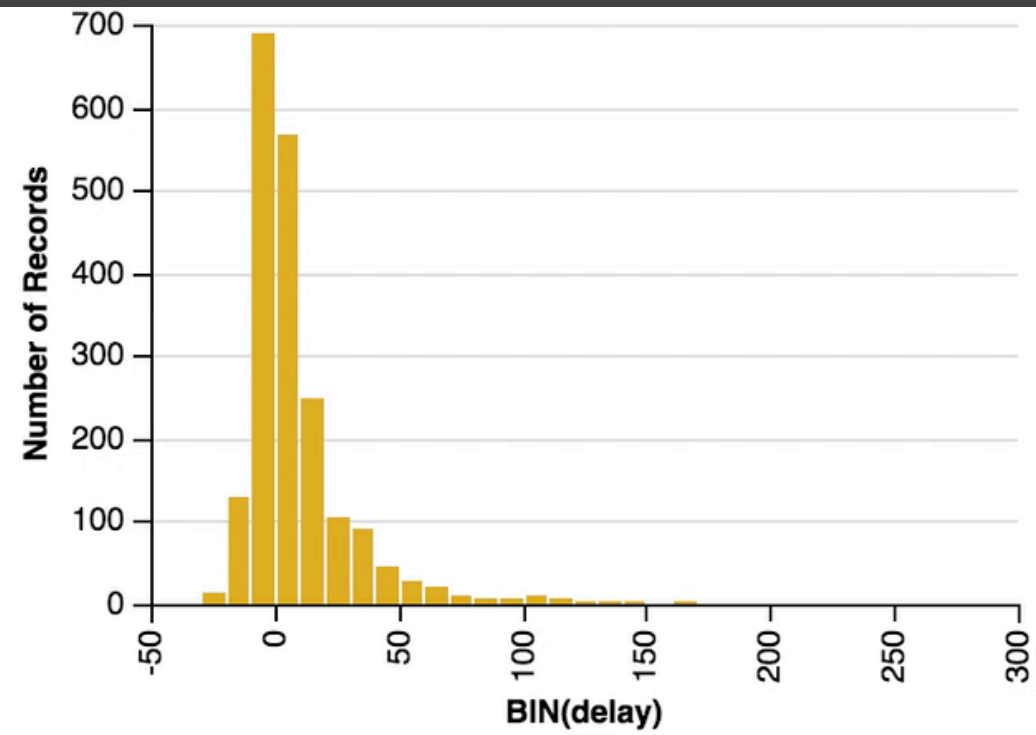
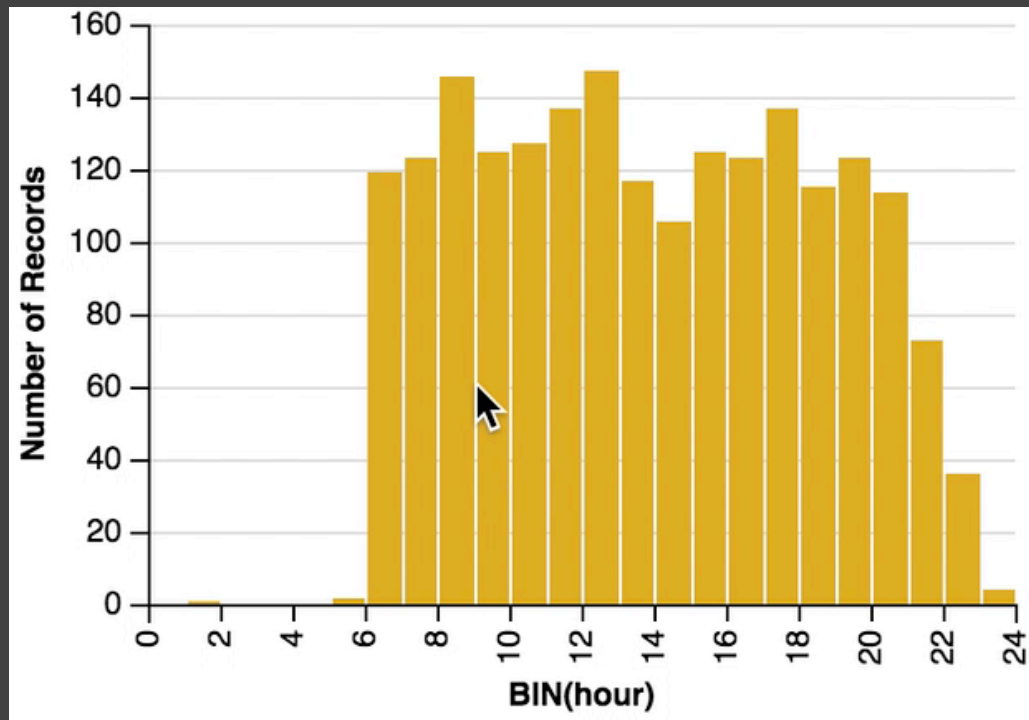


Focus + Context

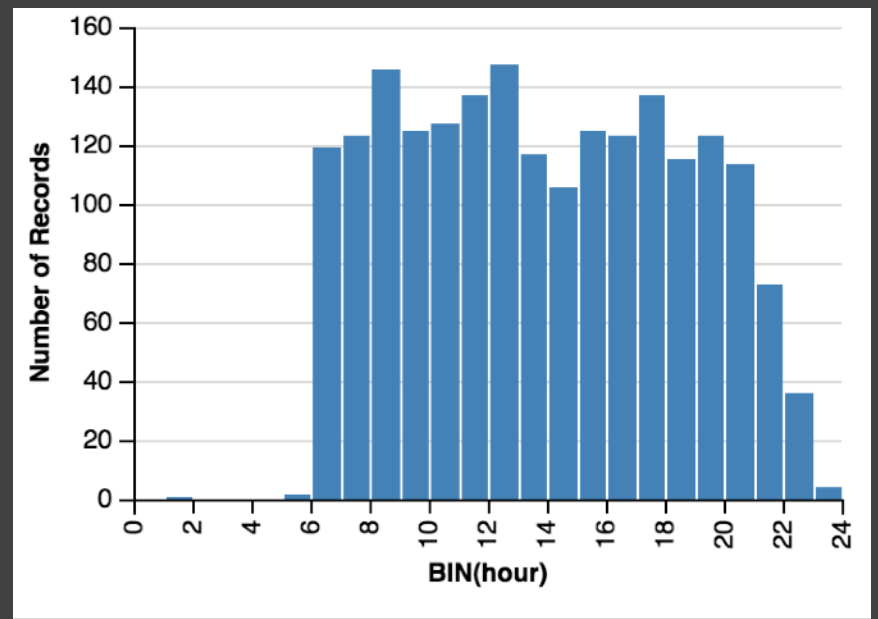


Cross-Filtering

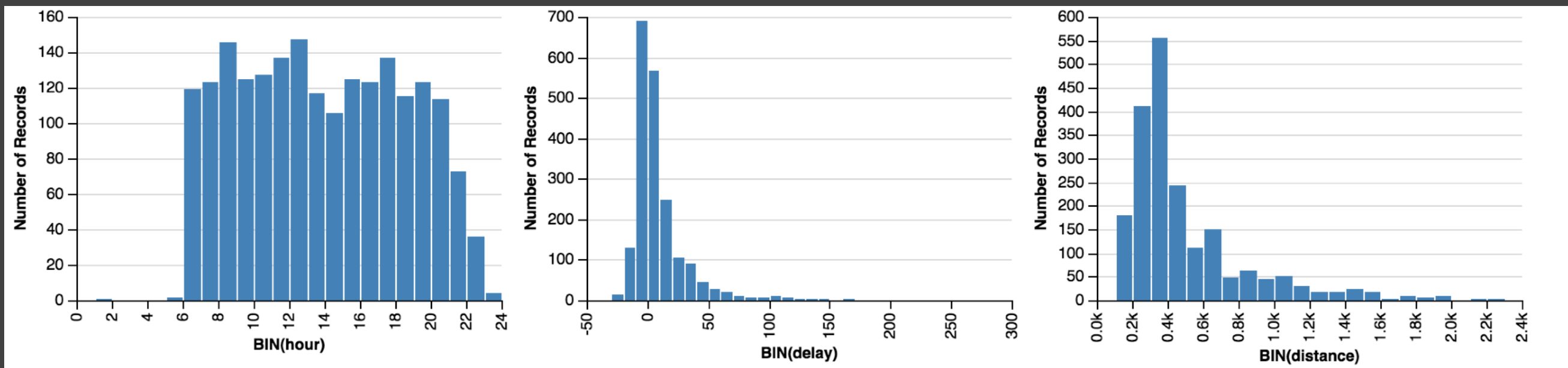
Vega-Lite: A Grammar of **Interactive** Multi-View Graphics



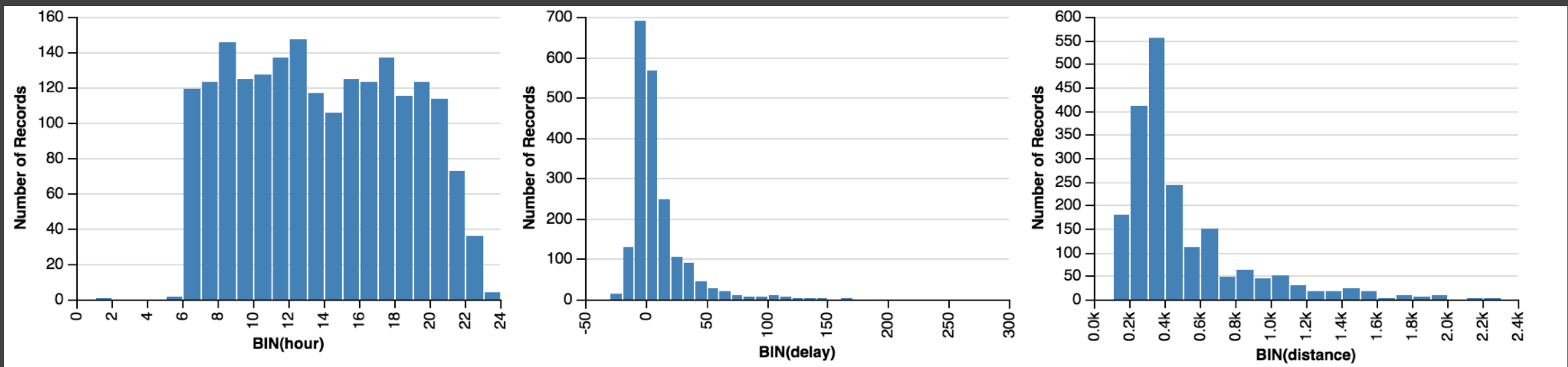
```
{
  "data": {"url": "data/flights.json"},
  "mark": "bar",
  "encoding": {
    "x": {"field": "hour", "bin": true, "type": "Q"},
    "y": {"field": "*", "aggregate": "count", "type": "Q"}
  }
}
```



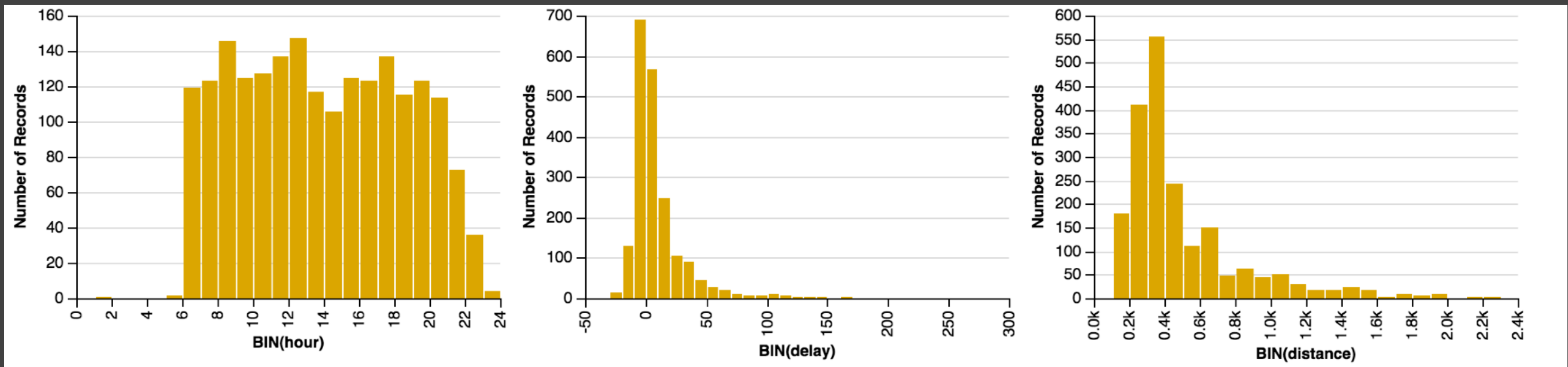
```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "data": {"url": "data/flights.json"},
    "mark": "bar",
    "encoding": {
      "x": {"field": {"repeat": "column"}, "bin": true, "type": "Q"},
      "y": {"field": "*", "aggregate": "count", "type": "Q"}
    }
  }
}
```



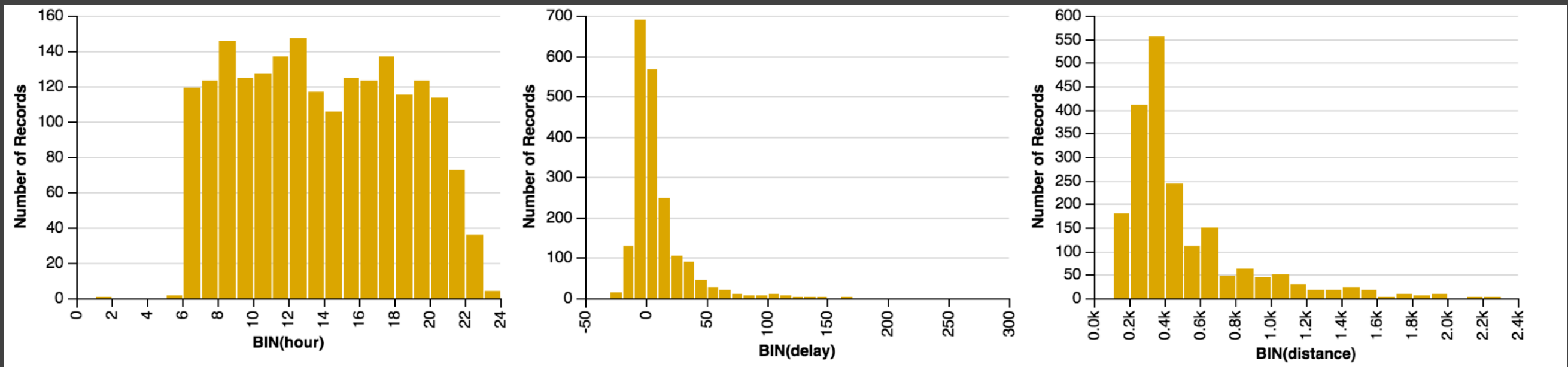
```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "data": {"url": "data/flights.json"},
    "mark": "bar",
    "encoding": {
      "x": {"field": {"repeat": "column"}, "bin": true, "type": "Q"},
      "y": {"field": "*", "aggregate": "count", "type": "Q"}
    }
  }
}
```



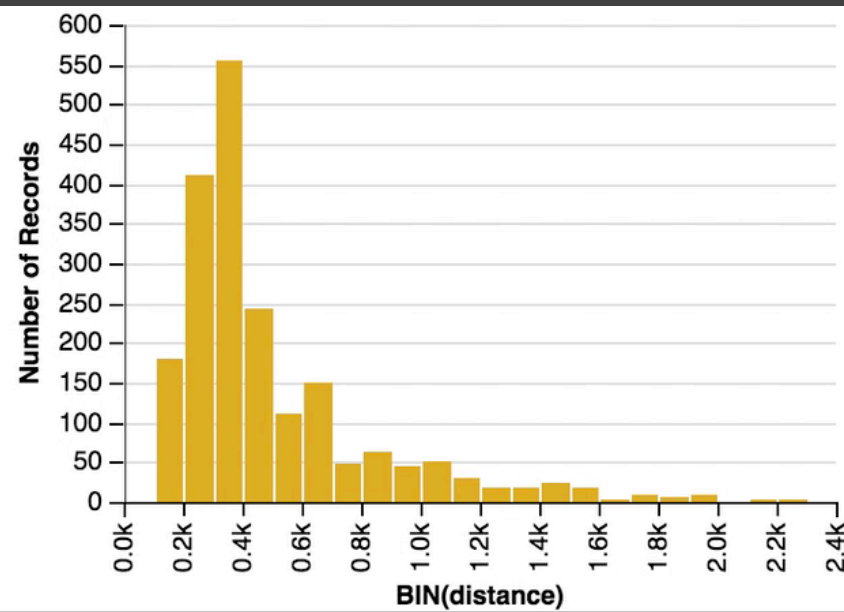
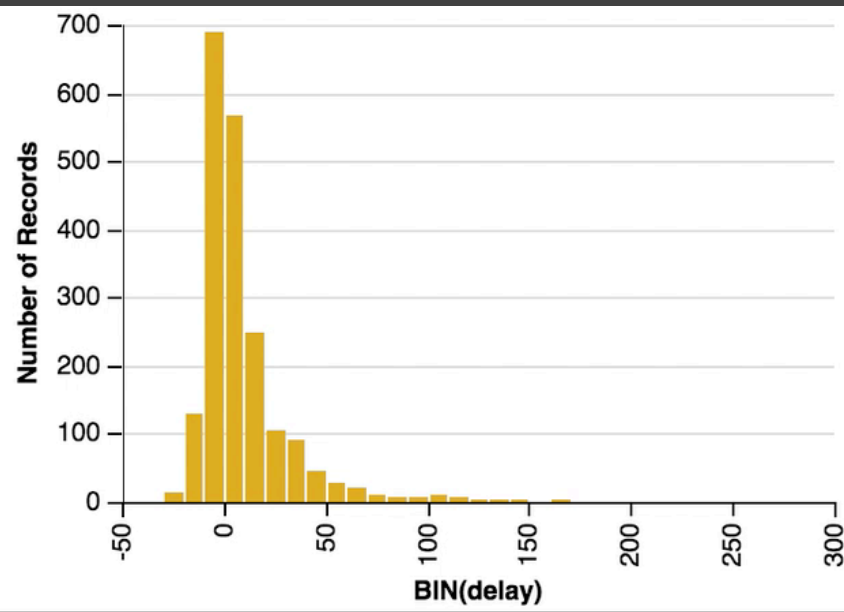
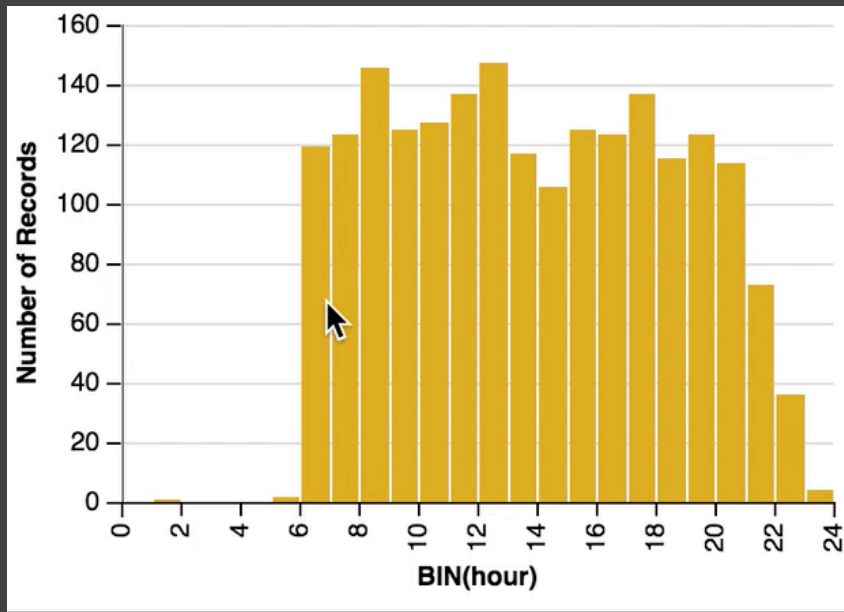
```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [{
      "data": {"url": "data/flights.json"},
      "mark": "bar",
      "encoding": {
        "x": {"field": {"repeat": "column"}, "bin": true, "type": "Q"},
        "y": {"field": "*", "aggregate": "count", "type": "Q"}
      }
    }],
    "encoding": {
      "color": {"value": "goldenrod"}
    }
  ]
}
```



```
{  
  "repeat": {"column": ["hour", "delay", "distance"]},  
  "spec": {  
    "layers": [  
      ...,  
  
    ], {  
      ...,  
  
    }]  
}
```

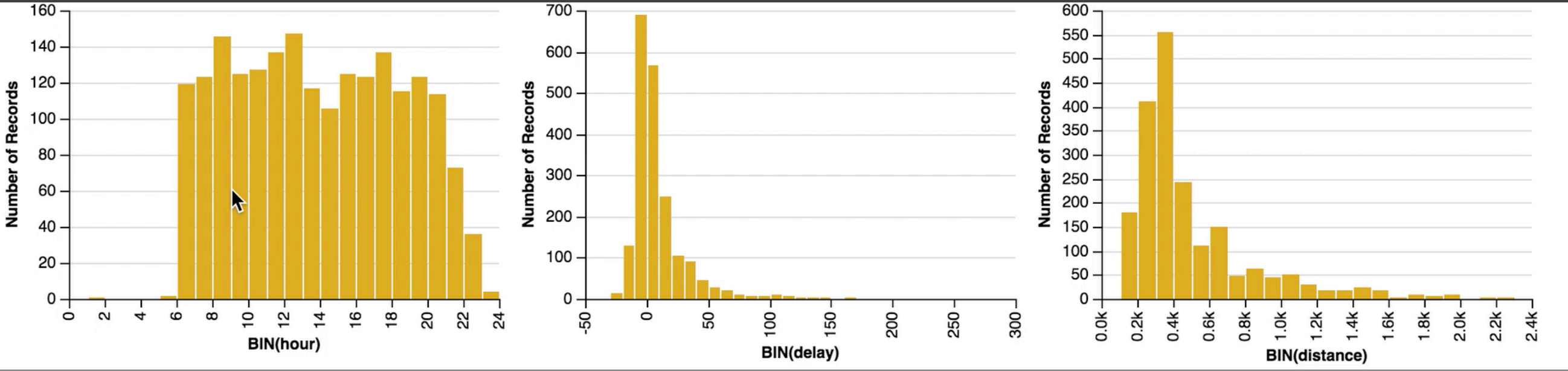



```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [
      ...,
      "select": {
        "region": {
          "type": "interval", "project": {"channels": ["x"]}, ...
        }
      }
    ], {
      ...,
    }
  }
}
```

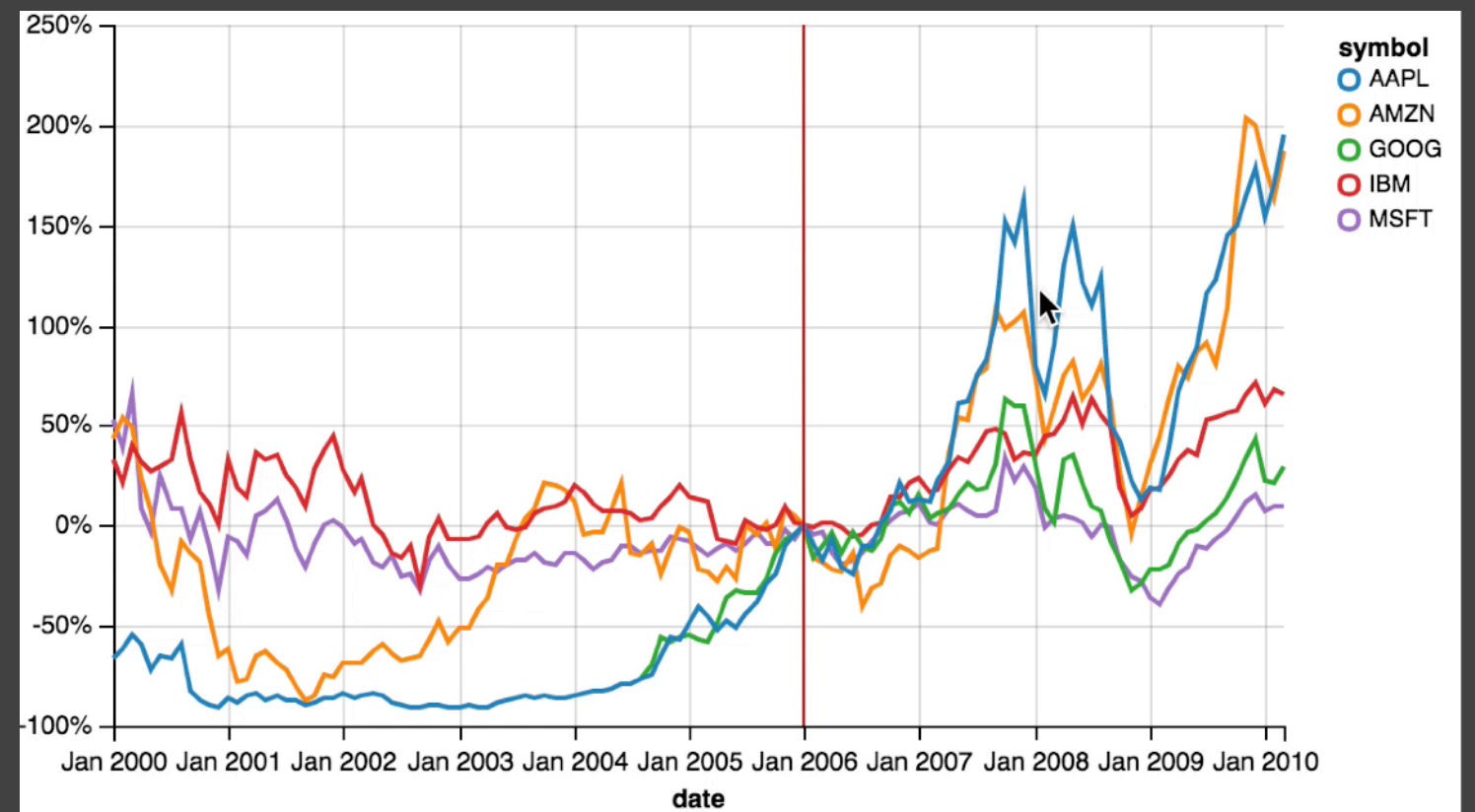
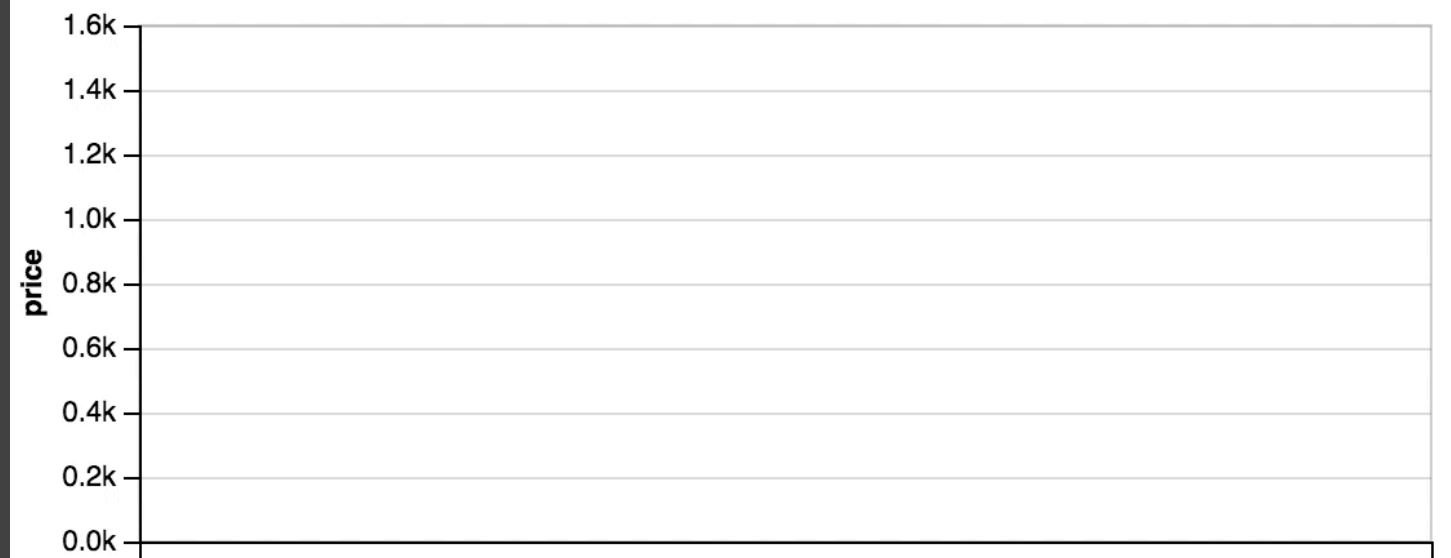
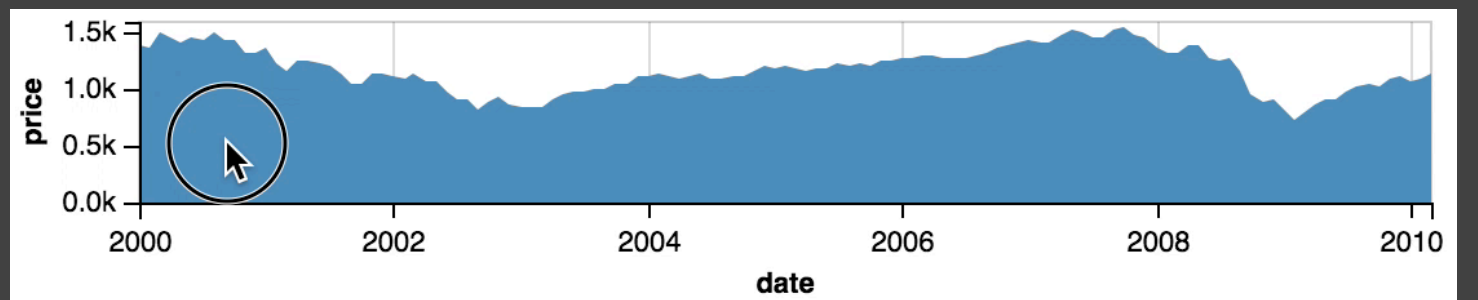
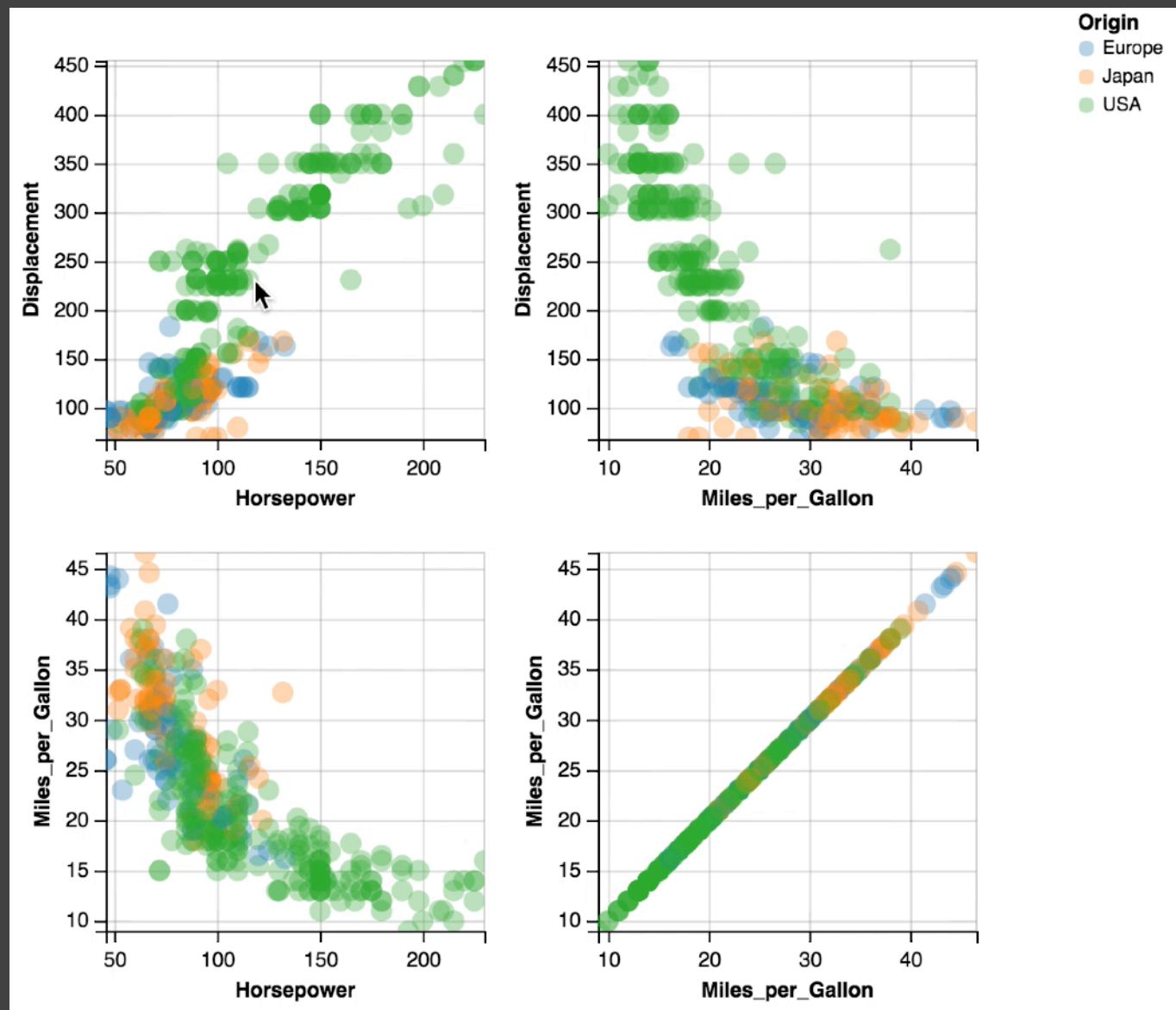


```
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [
      ...,
      "select": {
        "region": {
          "type": "interval", "project": {"channels": ["x"]}, ...
        }
      }
    ], {
      ...,
      "transform": {"filterWith": "region"}
    }
  ]
}
```

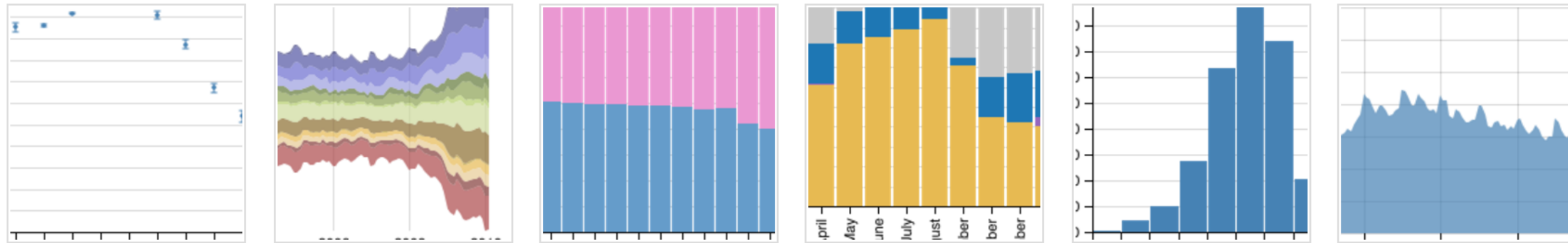
35 Lines
of JSON!



Interactive Selections



Declarative Visualization in Python



Altair is a declarative statistical visualization library for Python, based on **Vega-Lite**.

With Altair, you can spend more time understanding your data and its meaning. Altair's API is simple, friendly and consistent and built on top of the powerful **Vega-Lite** visualization grammar. This elegant simplicity produces beautiful and effective visualizations with a minimal amount of code.

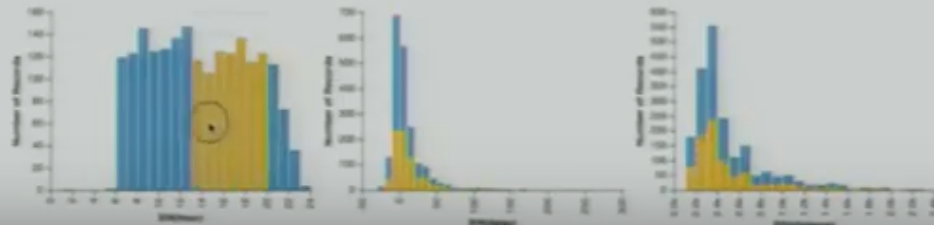
Altair: Vega-Lite in Python

Led by Jake VanderPlas & Brian Granger



Vega-Lite Layered Cross Filtering

```
{
  repeat: (column: ["hour", "delay", "distance"]),
  spec: {
    layer: [(
      ....
      selection: {
        region: (type: "interval", encodings: ["x"])
      }
    ), (
      ....
      transform: [(filter: (selection: "region"))]
    )
  ]
}
```



OPEN
VIS 2017
CONF

Full screen

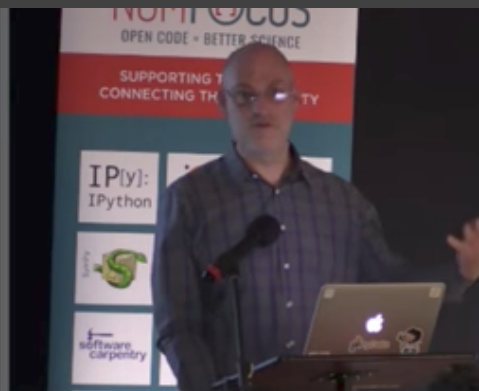
To Learn More...

Vega-Lite: A Grammar of Interactive Graphics, *OpenVis Conf 2017*

youtu.be/9uaHRWj04D4

Altair Example

```
In [1]: from altair import datasets, Chart
data = datasets.load_dataset('cars')
Chart(data).mark_circle().encode(
  x='Horsepower',
  y='Miles_per_Gallon',
  color='Origin',
)
```



PyData
SF 2016

PyData

Altair: Declarative Visualization for Python, *PyData SF 2016*

youtu.be/aRxahWy-ul8

How might we support more
effective data exploration?

Common analysis pitfalls:

Overlook data quality issues

Fixate on specific relationships

Plus many other cognitive biases

[Heuer 1999, Kahneman 2011, ...]

Voyager 2

Secure | <https://uwdata.github.io/voyager2/>

datavoyager

Bookmarks (0) Undo Redo

Data

Cars

Fields

- A Cylinders
- A Name
- A Origin
- Year
- # Acceleration
- # Displacement
- # Horsepower
- # Miles per Gallon
- # Weight in lbs
- # COUNT

Wildcards

- A Categorical Fields
- Temporal Fields
- # Quantitative Fields

Encoding

x

y

column

row

Marks

size

color

shape

detail

text

any

Filter

Related Views

Add Categorical Field

#

#

MEAN(Miles_per_Gallon)

YEAR(Year)

Cylinders

- 3
- 4
- 5
- 6
- 8

Origin

- Europe
- Japan
- USA

Debug · Report an issue

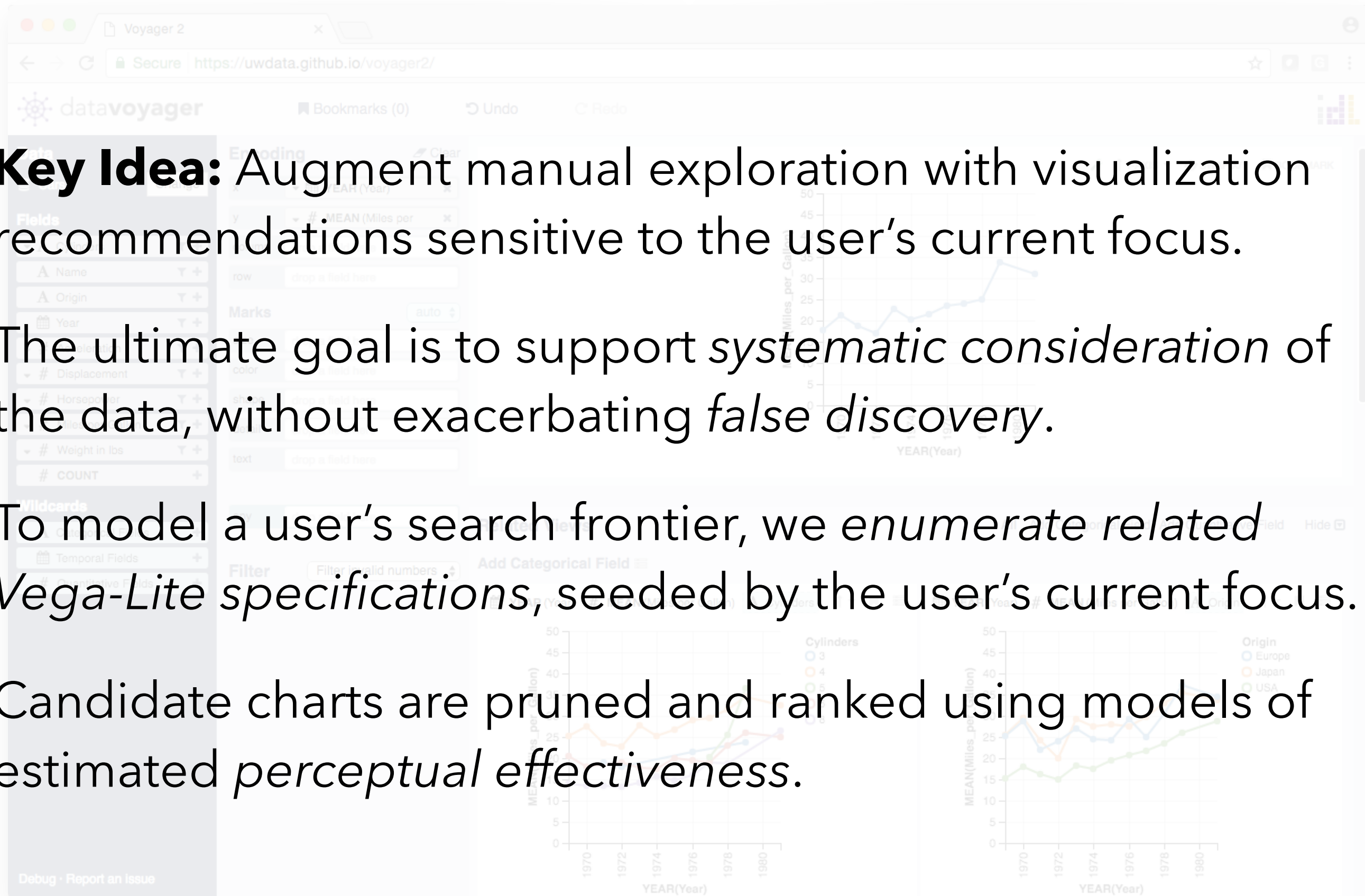
Voyager: Combine Manual Specification with Visualization Recommenders

Key Idea: Augment manual exploration with visualization recommendations sensitive to the user's current focus.

The ultimate goal is to support *systematic consideration* of the data, without exacerbating *false discovery*.

To model a user's search frontier, we *enumerate related Vega-Lite specifications*, seeded by the user's current focus.

Candidate charts are pruned and ranked using models of estimated *perceptual effectiveness*.



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

"The 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."