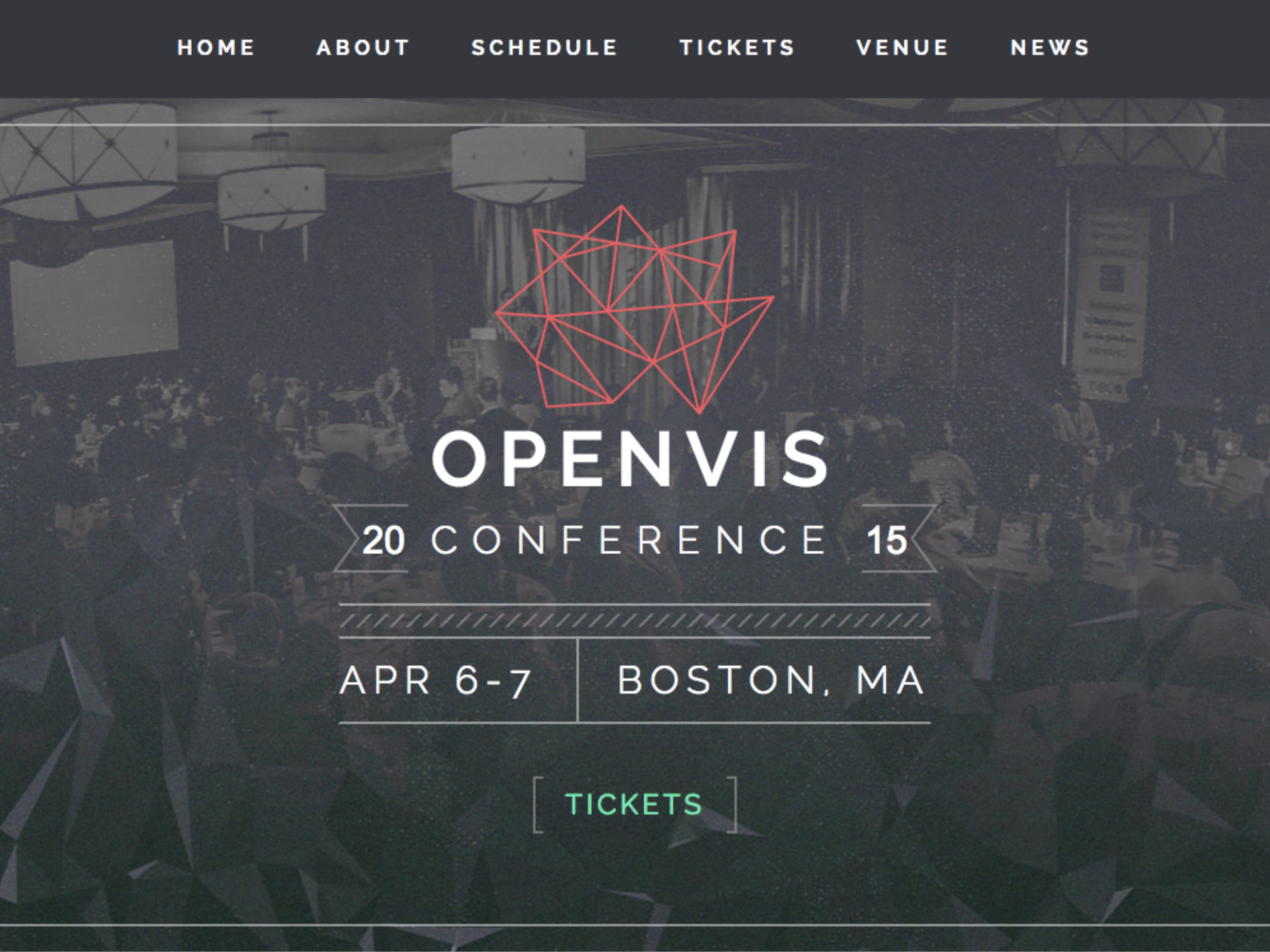
Raising the Bar (Chart)

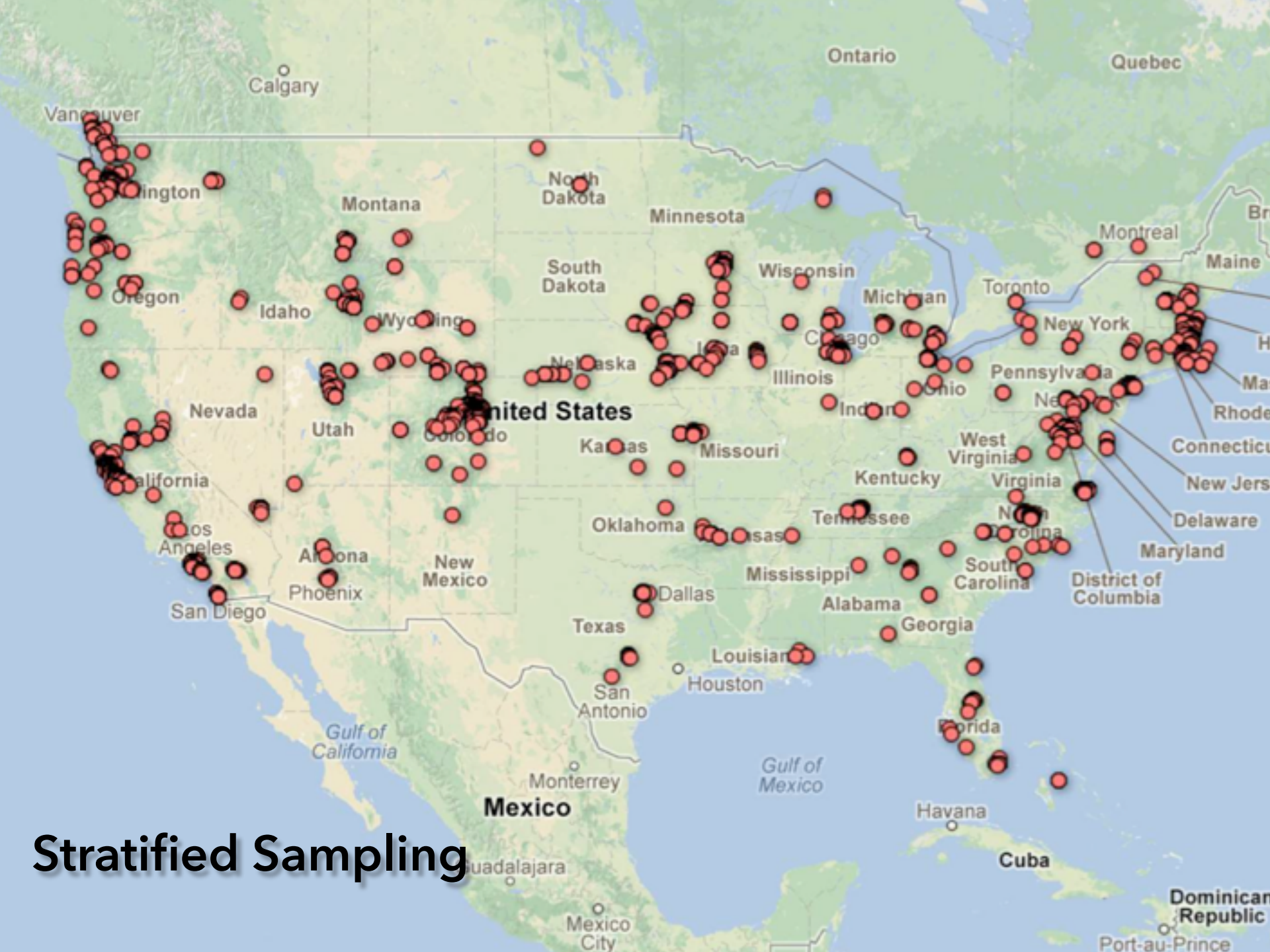
THE NEXT GENERATION OF VISUALIZATION TOOLS

Jeffrey Heer @jeffrey_heer
Univ. of Washington + Trifacta

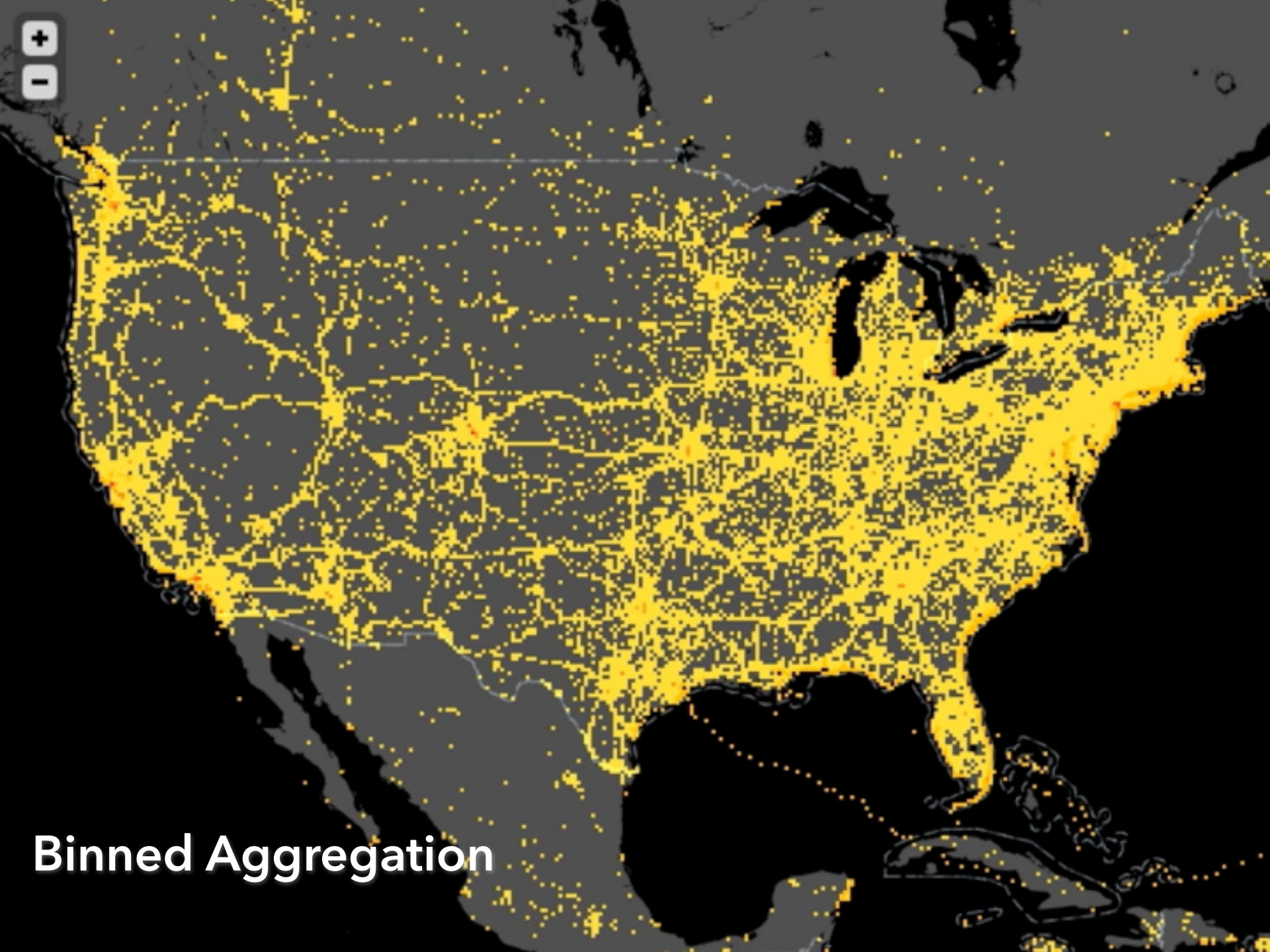

HOME
ABOUT
SCHEDULE
TICKETS
VENUE
NEWS
OPENVIS
20 CONFERENCE 15
APR 6-7
BOSTON, MA
TICKETS

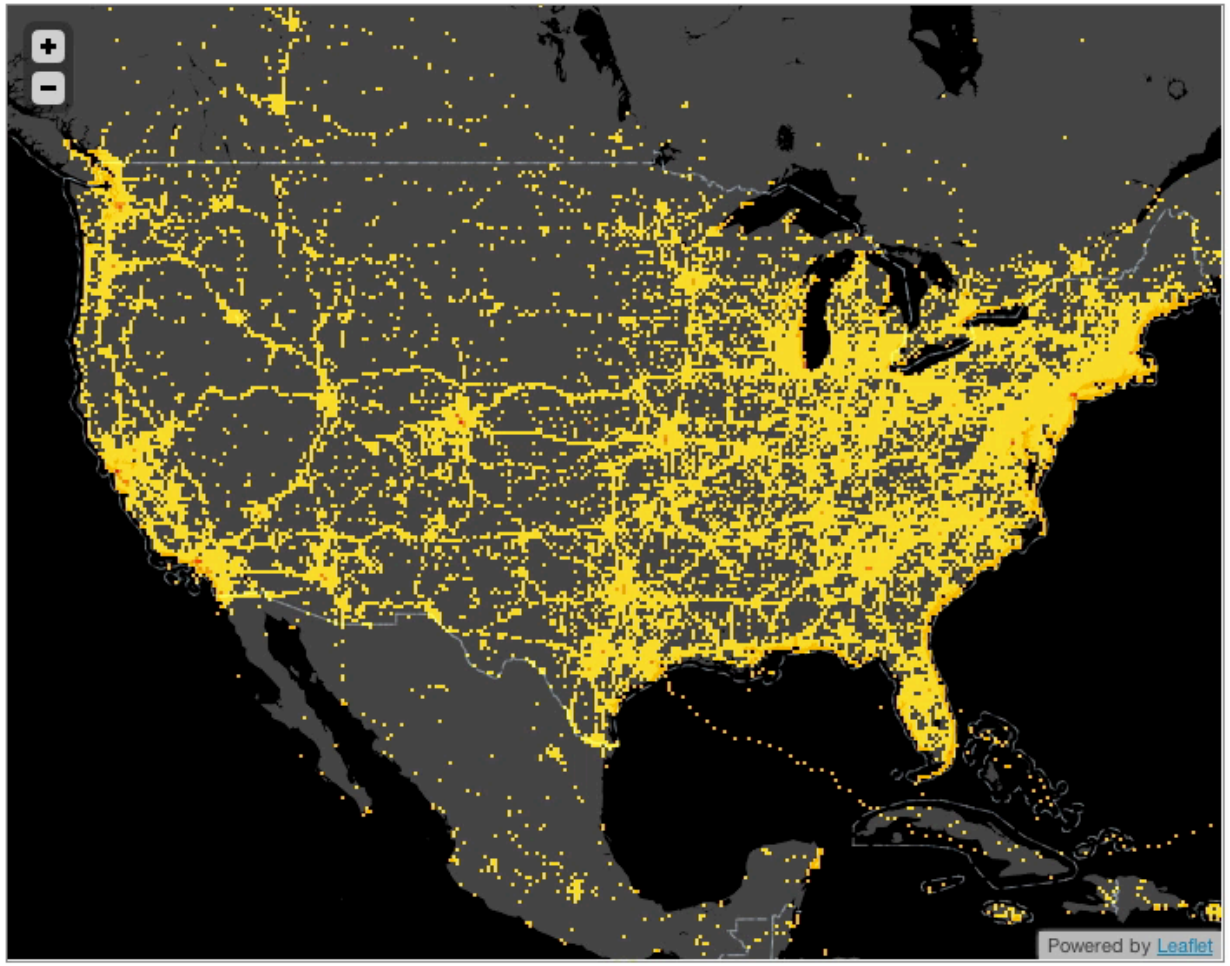

HOME
ABOUT
SCHEDULE
TICKETS
VENUE
NEWS
?
20 CONFERENCE 15
APR 6-7
BOSTON, MA
TICKETS

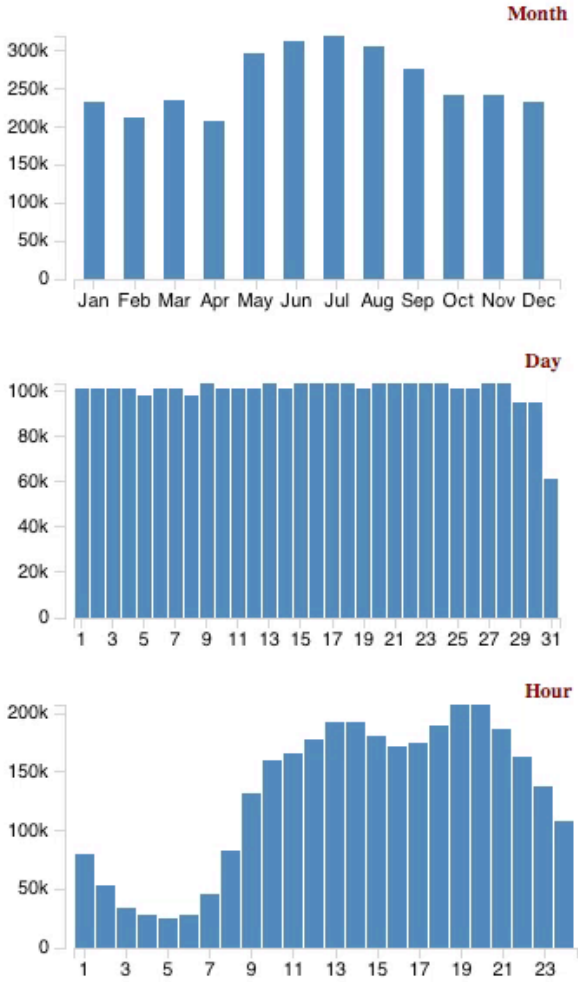
Visualizing Big Data!



Stratified Sampling


Binned Aggregation





imMens: Real-Time Visual Querying of Big Data

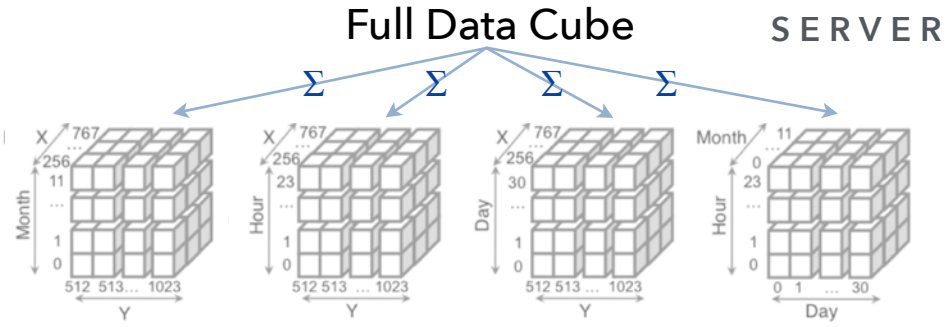
with **Z. Liu** & B. Jiang

2:15 - 2:45

DANYEL FISHER

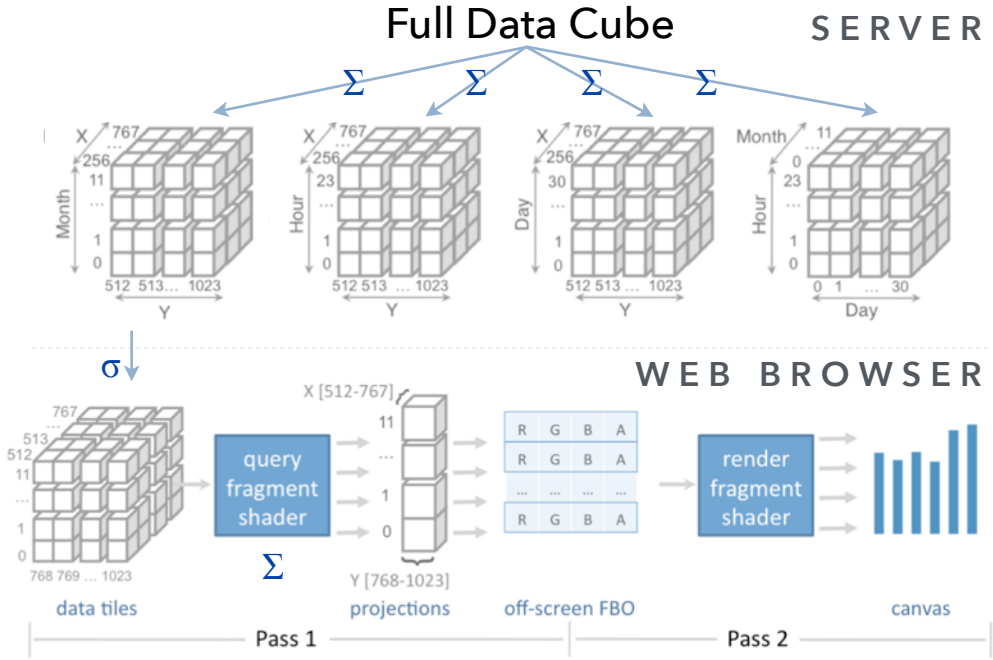
WHY EXPLORING BIG DATA IS HARD (AND WHAT WE CAN DO ABOUT IT)

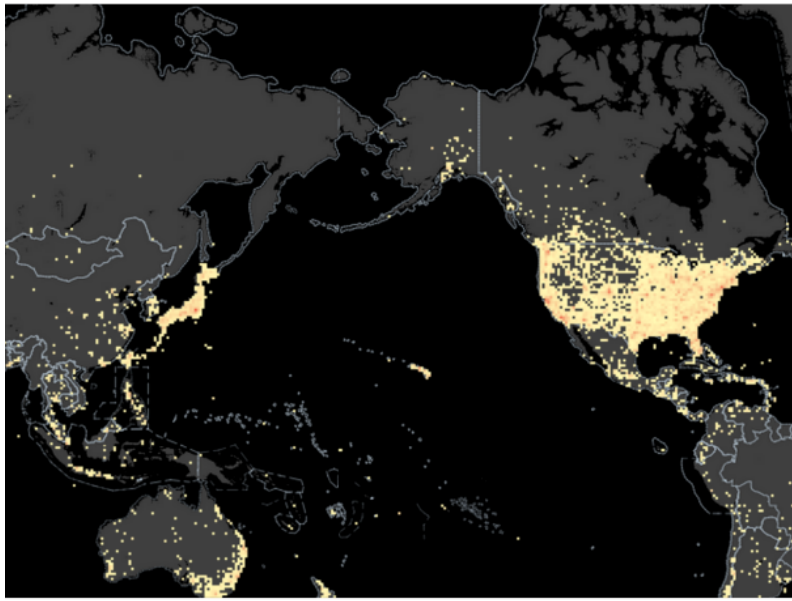
imMens - Real-time Querying of Big Data

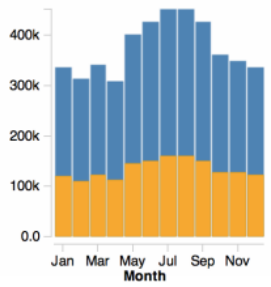


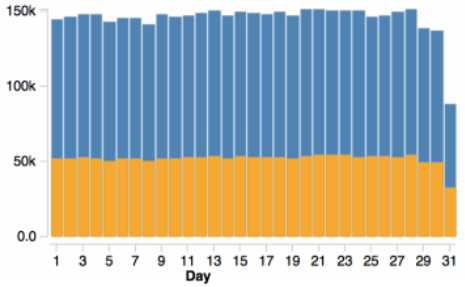
with **Z. Liu**, B. Jiang [EuroVis '13]

imMens - Real-time Querying of Big Data



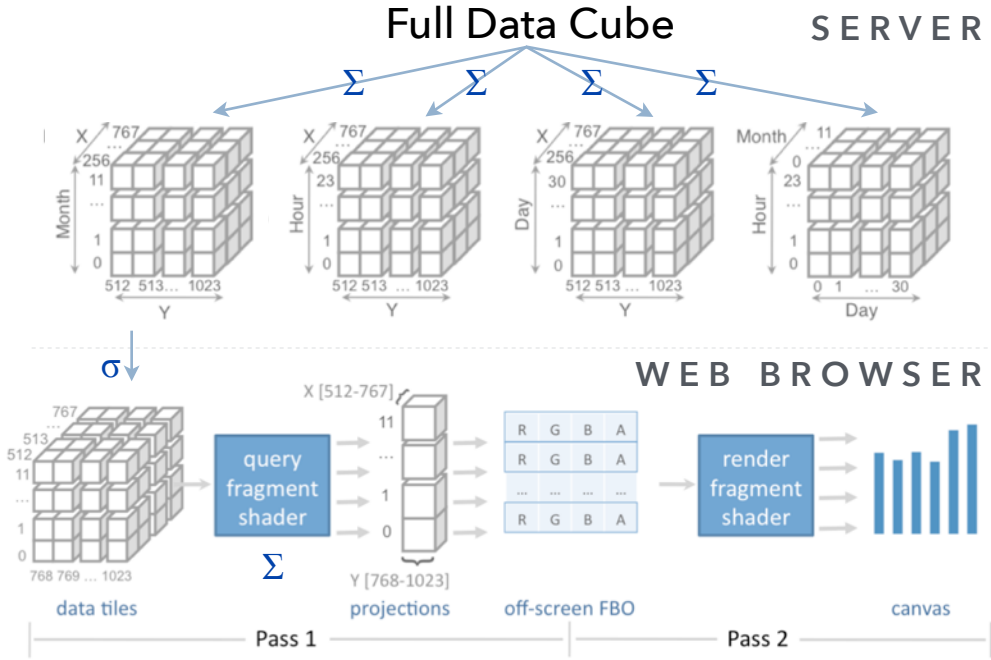


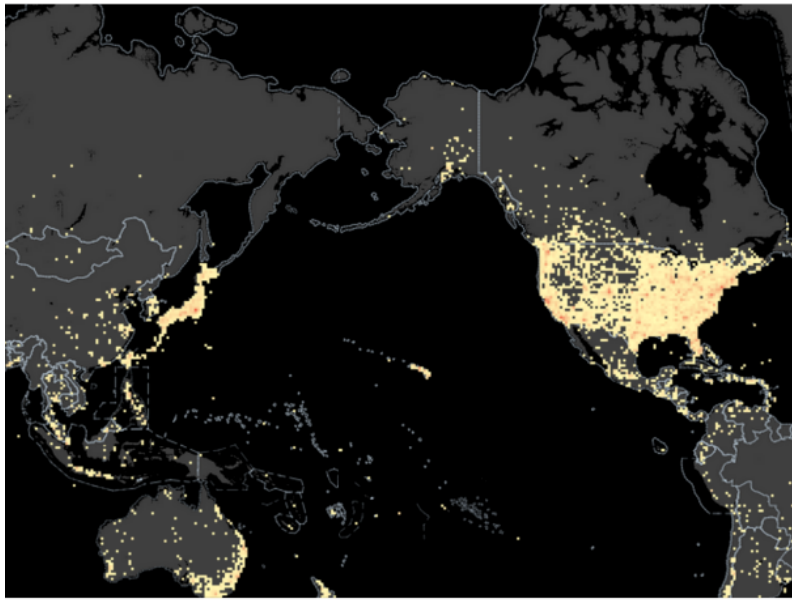


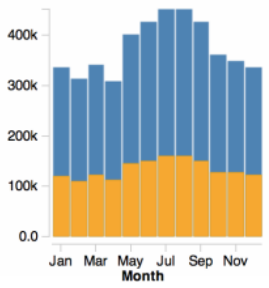


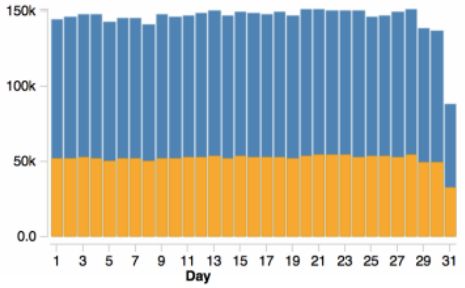
with **Z. Liu**, B. Jiang [EuroVis '13]

imMens - Real-time Querying of Big Data









50 fps interactive querying of summary visualizations of **billions of data points**.

with **Z. Liu**, B. Jiang [EuroVis '13]

WebGL!

1:30 - 2:15

NICOLAS GARCIA BELMONTE

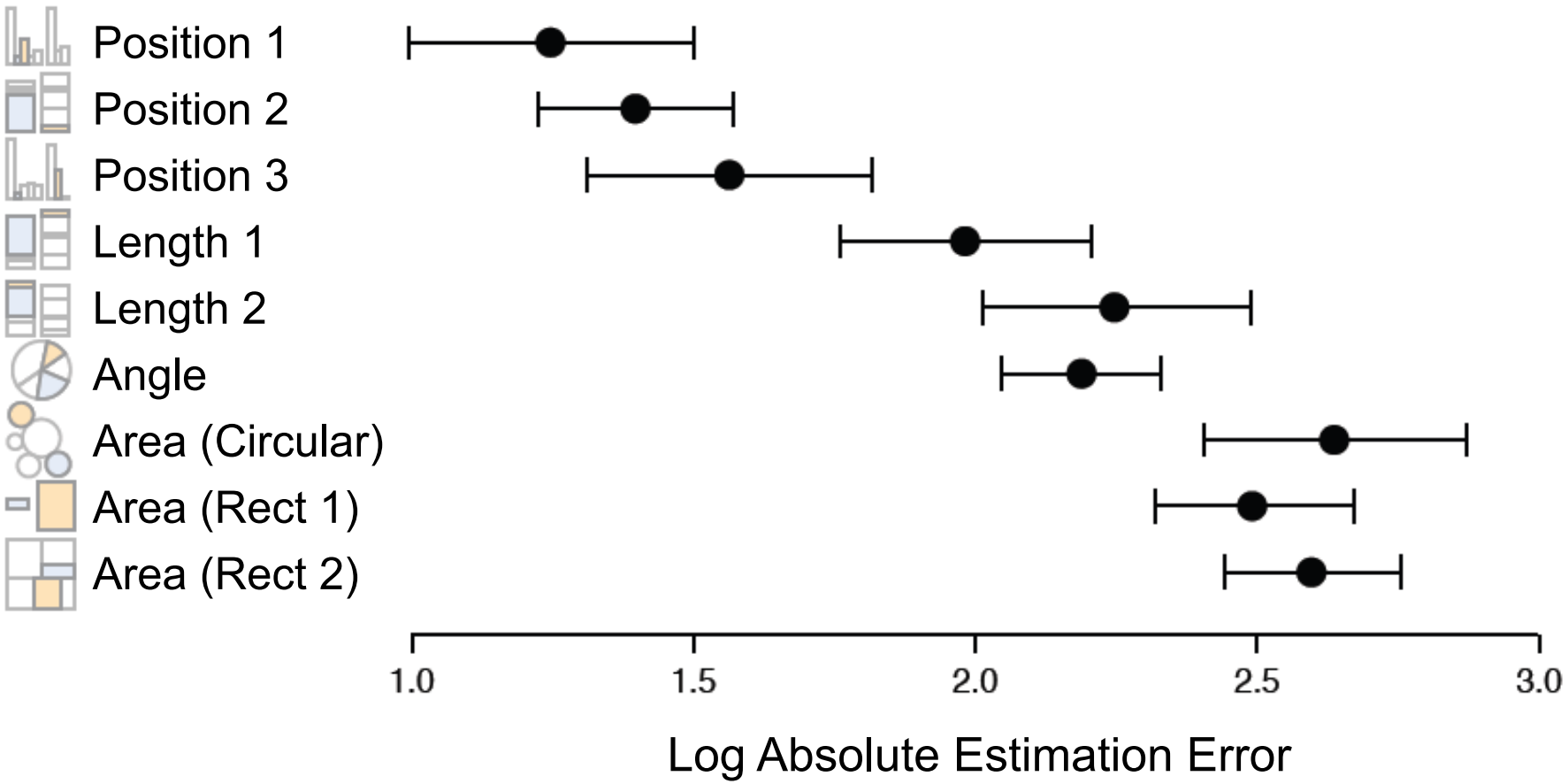
WEBGL FOR GRAPHICS AND DATA VISUALIZATION

1:30 - 2:15

DOMINIKUS BAUR

WEIGHING PERFORMANCE AGAINST PAIN: SVG, CANVAS AND WEBGL

Perception!



Graphical Perception [CHI'09, CHI'10, InfoVis'10…]

Experiments to assess the effectiveness of visual encodings.

Cognitive Color Models [CHI '12, EuroVis '13]

Novel measures informing color-concept maps & palette design.

12:00 - 12:30

LANE HARRISON

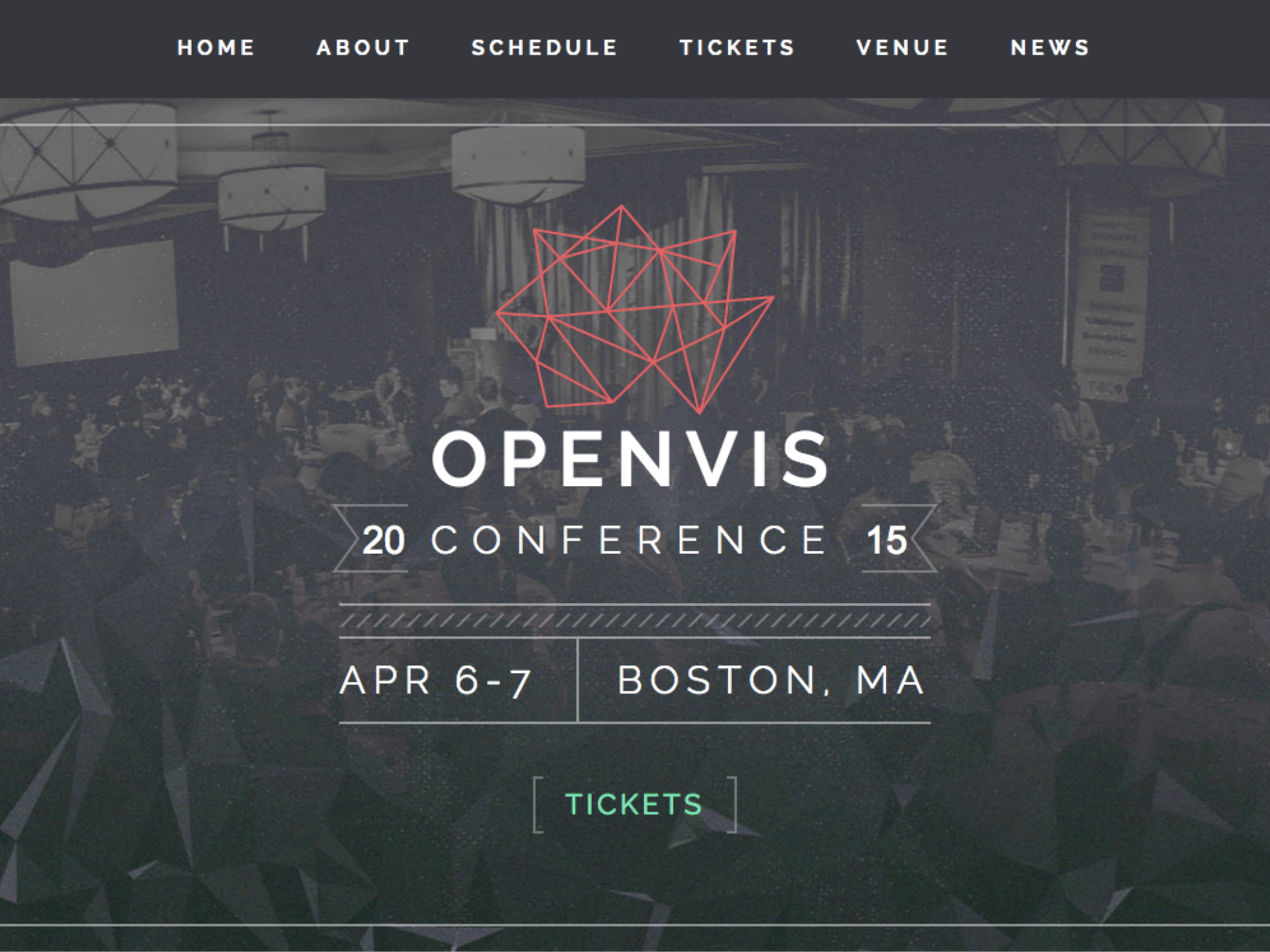
USER-CENTERED
VISUALIZATION RESEARCH


HOME
ABOUT
SCHEDULE
TICKETS
VENUE
NEWS
???
OPENVIS
20 CONFERENCE 15
APR 6-7
BOSTON, MA
TICKETS

3:00 - 3:30

NIGEL HOLMES

USING HUMOR TO INFORM


HOME
ABOUT
SCHEDULE
TICKETS
VENUE
NEWS
OPENVIS
20 CONFERENCE 15
APR 6-7
BOSTON, MA
TICKETS

Visualization Tools

prefuse

Protovis

vega

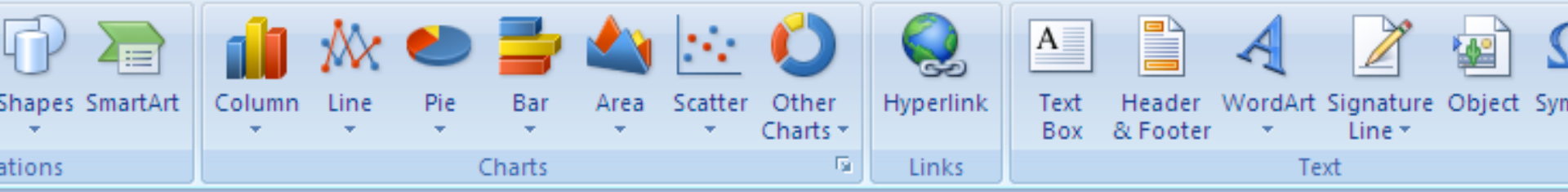
Data-Driven Documents

Raising the Bar (Chart)

THE NEXT GENERATION OF VISUALIZATION TOOLS

Jeffrey Heer @jeffrey_heer
Univ. of Washington + Trifacta

How might we
create visualizations?



fx L1

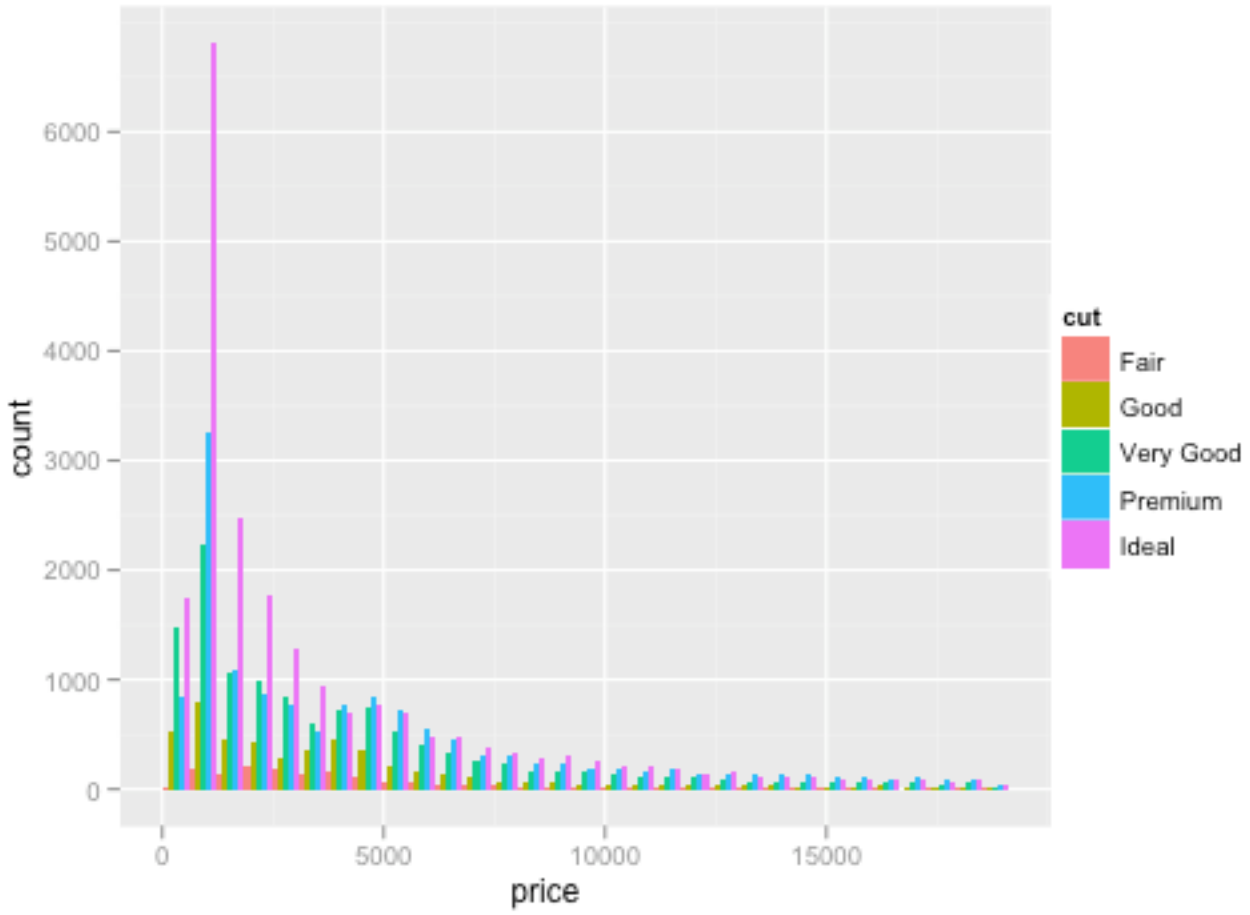
B	C	D	E	F	G	H	I	J
l	task	row	col	d	d1	d2	L1	L2
-color	SA	1	1	0	0	0	0	0
-color	SA	1	2	0.63512	0.73568	0	0.73568	0.73568
-color	SA	1	3	0.59295	0.73119	0	0.73119	0.73119
-color	SA	1	4	0.45864	0.24612	0	0.24612	0.24612
-color	SA	1	5	0.49399	0	0.8119	0.8119	0.8119
-color	SA	1	6	0.85906	0.73568	0.8119	1.54758	1.095630719
-color	SA	1	7	0.81534	0.73119	0.8119	1.54309	1.092620898
-color	SA	1	8	0.71854	0.24612	0.8119	1.05802	0.848384738
-color	SA	1	9	0.57947	0	0.8417	0.8417	0.8417
-color	SA	1	10	0.91546	0.73568	0.8417	1.57738	1.117892639
-color	SA	1	11	0.91988	0.73119	0.8417	1.57289	1.114942916
-color	SA	1	12	0.77741	0.24612	0.8417	1.08782	0.876945805
-color	SA	1	13	0.29483	0	0.2825	0.2825	0.2825
-color	SA	1	14	0.75357	0.73568	0.2825	1.01818	0.788055399
-color	SA	1	15	0.73206	0.73119	0.2825	1.01369	0.783865464
-color	SA	1	16	0.53588	0.24612	0.2825	0.52862	0.374674932

Layout Formulas Data Review View Add-Ins

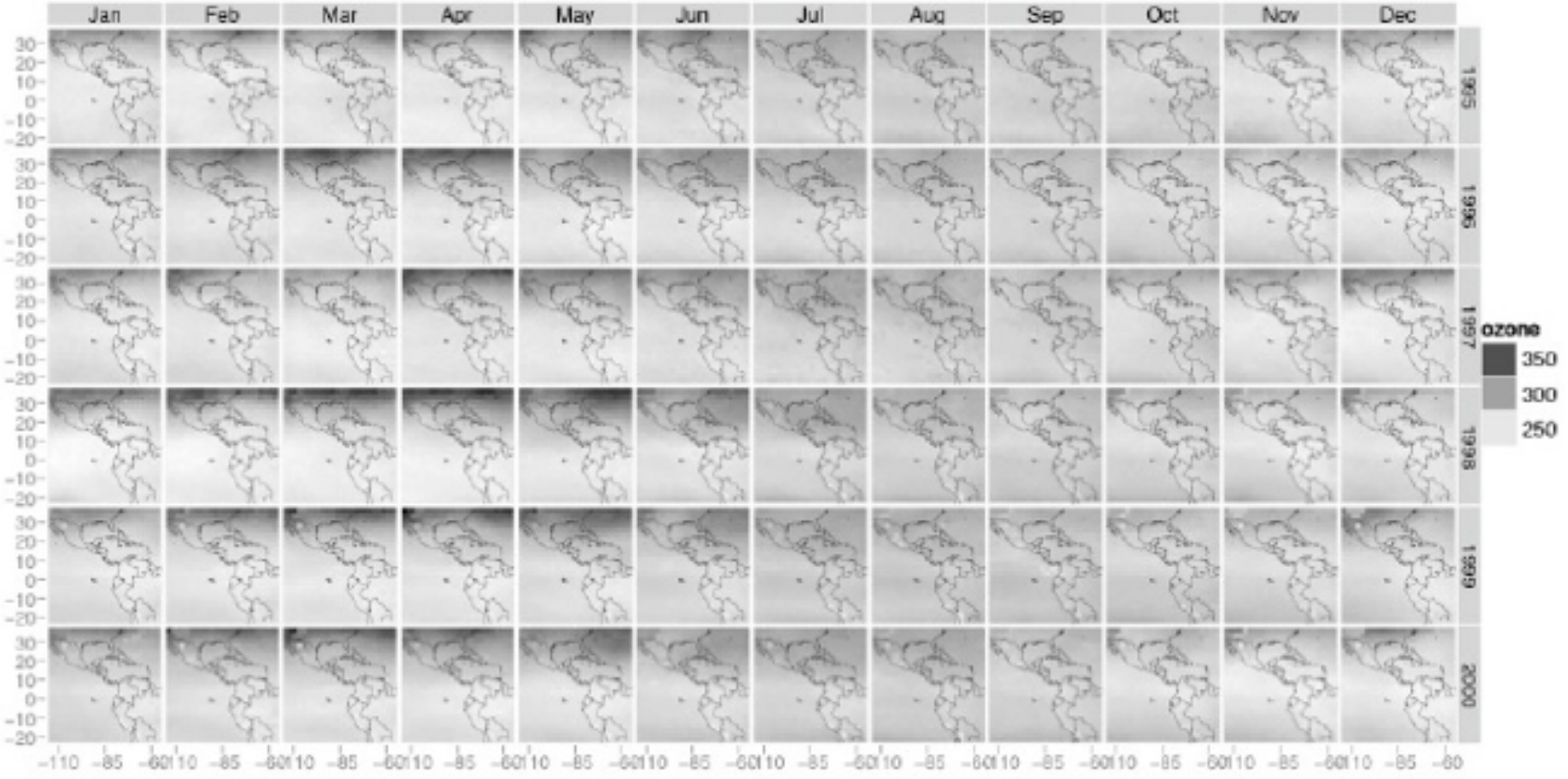
fx L1

B	C	D	H	I	J
	task	row	d2	L1	L2
-color	SA	1	0	0	0
-color	SA	1	0	0.73568	0.73568
-color	SA	1	0	0.73119	0.73119
-color	SA	1	0	0.24612	0.24612
-color	SA	1	0.8119	0.8119	0.8119
-color	SA	1	0.8119	1.54758	1.095630719
-color	SA	1	0.8119	1.54309	1.092620898
-color	SA	1	0.8119	1.05802	0.848384738
-color	SA	1	0.8417	0.8417	0.8417
-color	SA	1	0.8417	1.57738	1.117892639
-color	SA	1	0.8417	1.57289	1.114942916
-color	SA	1	0.8417	1.08782	0.876945805
-color	SA	1	0.2825	0.2825	0.2825
-color	SA	1	0.2825	1.01818	0.788055399
-color	SA	1	0.2825	1.01369	0.783865464
-color	SA	1	0.2825	0.52862	0.374674932
		16 0.53588 0.24612			

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




```
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
```

```
var svg = div.append("svg:svg")
    .attr("width", w)
    .attr("height", w)
  .append("svg:g")
    .attr("transform", "translate(" + rx + "," + ry + ")");

svg.append("svg:path")
    .attr("class", "arc")
    .attr("d", d3.svg.arc().outerRadius(ry - 120).innerRadius(0).startAngle(0).endAngle(2 * Math.PI))
    .on("mousedown", mousedown);

d3.json("data/flare-imports.json", function(classes) {
  var nodes = cluster.nodes(packages.root(classes)),
      links = packages.imports(nodes),
      splines = bundle(links);

  var path = svg.selectAll("path.link")
      .data(links)
    .enter().append("svg:path")
      .attr("class", function(d) { return "link source-" + d.source.key + " target-" + d.target.key; })
      .attr("d", function(d, i) { return line(splines[i]); });

  svg.selectAll("g.node")
      .data(nodes.filter(function(n) { return !n.children; }))
    .enter().append("svg:g")
      .attr("class", "node")
      .attr("id", function(d) { return "node-" + d.key; })
      .attr("transform", function(d) { return "rotate(" + (d.x - 90) + ")translate(" + d.y + ")"; })
    .append("svg:text")
      .attr("dx", function(d) { return d.x < 180 ? 8 : -8; })
      .attr("dy", ".31em")
      .attr("text-anchor", function(d) { return d.x < 180 ? "start" : "end"; })
      .attr("transform", function(d) { return d.x < 180 ? null : "rotate(180)"; })
      .text(function(d) { return d.key; })
      .on("mouseover", mouseover)
      .on("mouseout", mouseout);

  d3.select("input[type=range]").on("change", function() {
    line.tension(this.value / 100);
    path.attr("d", function(d, i) { return line(splines[i]); });
  });
});
```

Graphics APIs

Processing, OpenGL, Java2D

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

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2

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

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

Visualization Grammars

Protovis, D3.js

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Programming Toolkits

Chart Typologies

Excel, Many Eyes, Google Charts

Charting Tools

Visual Analysis Grammars

VizQL, ggplot2

Visualization Grammars

Protovis, D3.js

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Programming Toolkits

What is a Declarative Language?

What is a Declarative Language?

Programming by describing *what*, not *how*

What is a Declarative Language?

Programming by describing *what*, not *how*

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

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*

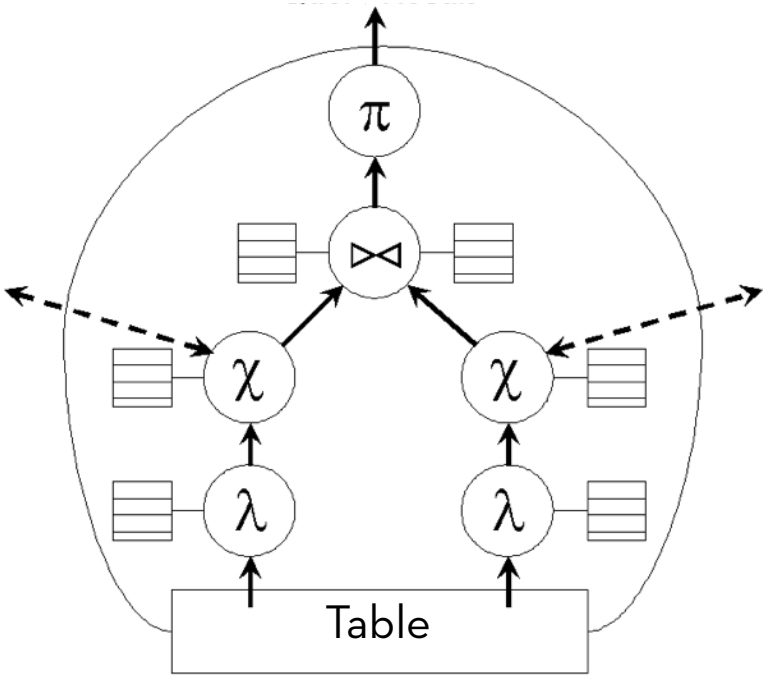
Separate **specification** (*what you want*) from **execution** (*how it should be computed*)

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

```
d3.selectAll("rect")
  .data(my_data)
 .enter().append("rect")
  .attr("x", function(d) { return xscale(d.foo); })
  .attr("y", function(d) { return yscale(d.bar); })
```


```
<!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
    <img src="/adx/bin/clientside/
    1b1a1755Q2F,.tWdC,KiQ2BCQ5C6iSM.KCYQ2FMadYYQ24xN!Q2B!xQ24WN4aN
    <script src="http://graphics8.nytimes.com/js/test/wto/wt_capi.
    charset="utf-8"></script>
    <span id="wto_script"></span>
  ▶<script type="text/javascript">…</script>
    <img src="http://graphics8.nytimes.com/ads/blank.gif">
    <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_nam
HAVING COUNT(order_id) > 5
ORDER BY COUNT(order_id) DESC
```

SQL

Chart Typologies

Excel, Many Eyes, Google Charts

Charting Tools

Visual Analysis Grammars

VizQL, ggplot2

Visualization Grammars

Protovis, D3.js

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Programming Toolkits

Chart Typologies

Excel, Many Eyes, Google Charts

Charting Tools

Visual Analysis Grammars

VizQL, ggplot2, ***Vegalite***

Visualization Grammars

Protovis, D3.js, ***Vega***

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Programming Toolkits

Why Declarative Languages?

Why Declarative Languages?

Faster iteration. Less code. Larger user base.

Why Declarative Languages?

Faster iteration. Less code. Larger user base.

Better visualization. *Smart defaults.*

Why Declarative Languages?

Faster iteration. Less code. Larger user base.

Better visualization. *Smart defaults.*

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

Why Declarative Languages?

Faster iteration. Less code. Larger user base.

Better visualization. *Smart defaults.*

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

Performance. *Optimization, scalability.*

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.*

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, ***Vegalite***

Visualization Grammars

Protovis, D3.js, ***Vega***

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Programming Toolkits

Chart Typologies

Excel, Many Eyes, Google Charts

?!

Charting Tools

Visual Analysis Grammars

VizQL, ggplot2, ***Vegalite***

Visualization Grammars

Protovis, D3.js, ***Vega***

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Programming Toolkits

Visual Analysis Grammars

VizQL, ggplot2, ***Vegalite***

Visualization Grammars

Protovis, D3.js, ***Vega***

Declarative Languages

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Programming Toolkits

Interactive Data Exploration

Tableau, ***Lyra, Polestar, Voyager***

Graphical Interfaces

Visual Analysis Grammars

VizQL, ggplot2, ***Vegalite***

Visualization Grammars

Protovis, D3.js, ***Vega***

Declarative Languages

Component Architectures

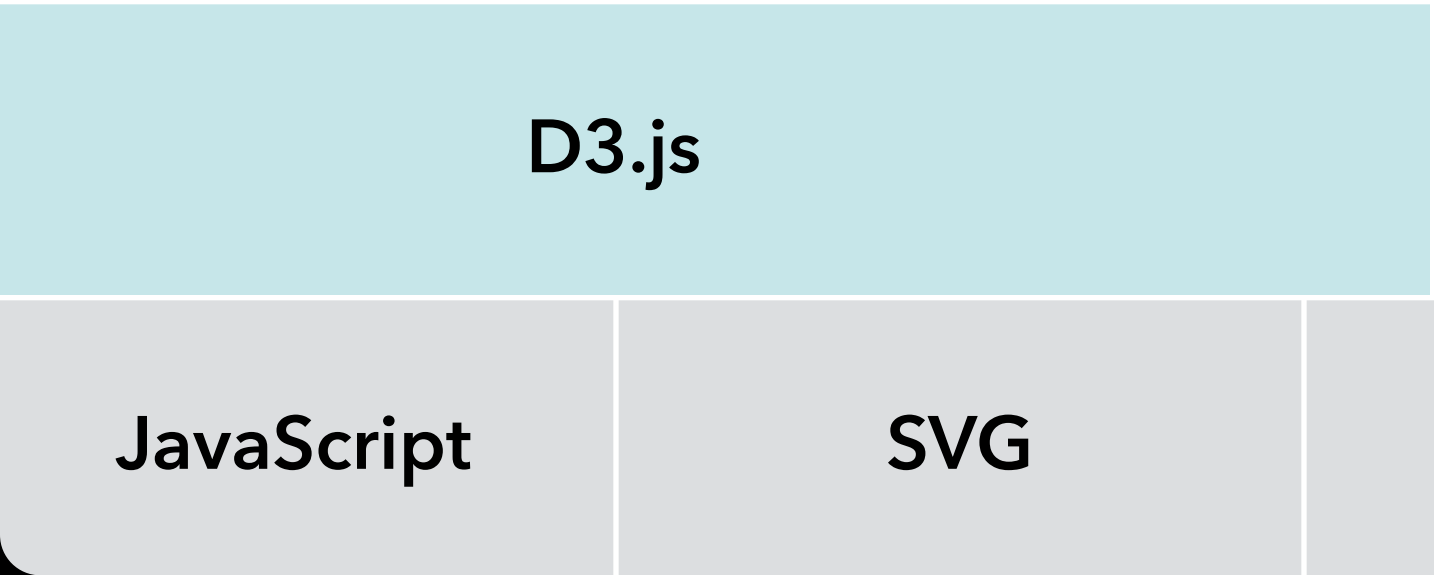
Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Programming Toolkits

JavaScript	SVG	Canvas


D3.js
JavaScript
SVG
Canvas


Vega
D3.js
JavaScript
SVG
Canvas

Visualization Grammar

Visualization Grammar

Data Input data to visualize

Visualization Grammar

Data	Input data to visualize
Transforms	Grouping, stats, projection, layout

Visualization Grammar

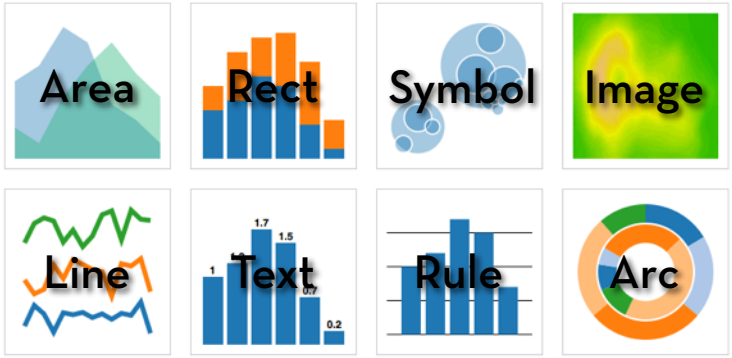
Data	Input data to visualize
Transforms	Grouping, stats, projection, layout
Scales	Map data values to visual values

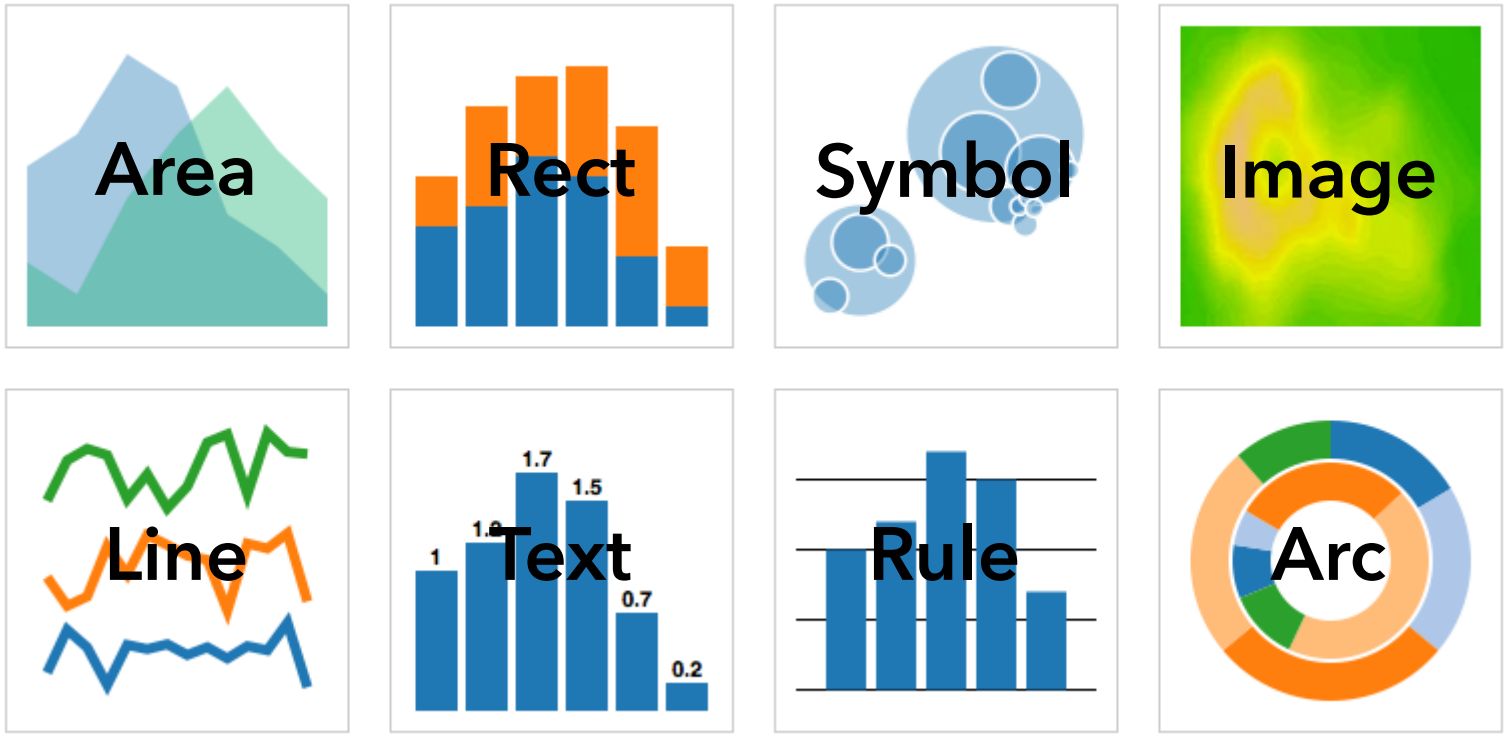
Visualization Grammar

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

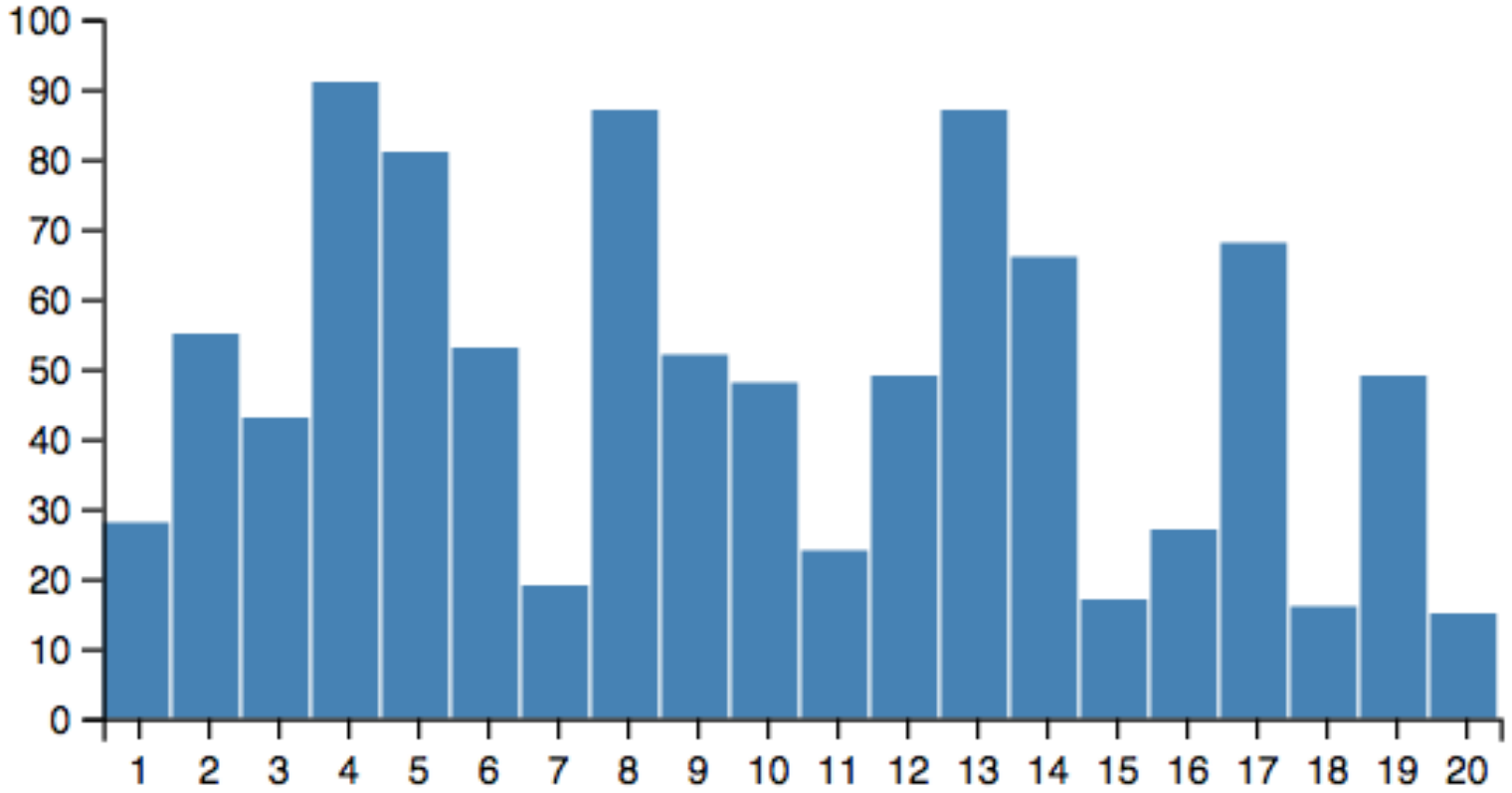
Visualization Grammar

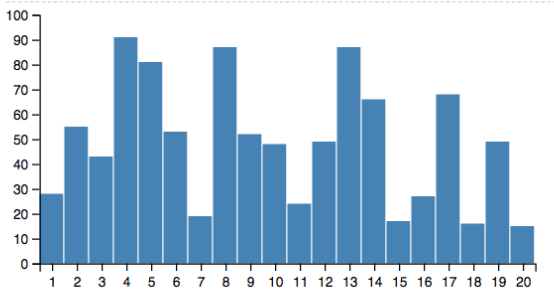
Data	Input data to visualize
Transforms	Grouping, stats, projection, layout
Scales	Map data values to visual values
Guides	Axes & legends visualize scales
Marks	Data-representative graphics



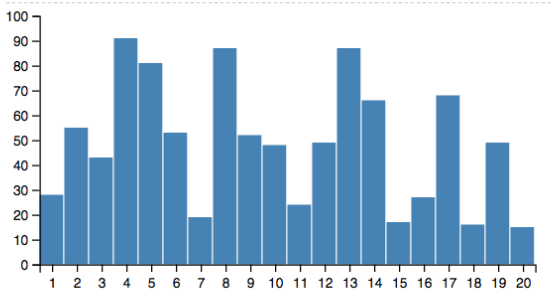


MARKS: Graphical Primitives


100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

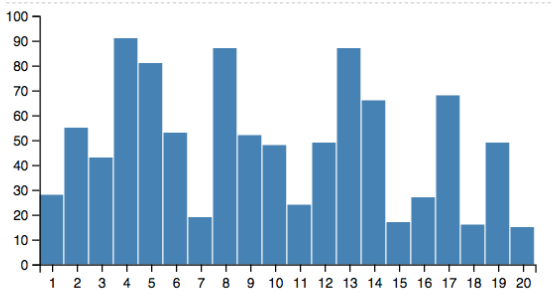

100
90
80
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

```
{
  "width": 400, "height": 200,
  "data": [
    {"name": "table", "url": "/data/sample.json"}
  ],
  "scales": [
    {
      "name": "x","type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "range": “height", "nice": true,
      "domain": {"data": "table", "field": "y"}
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [{
    "type": "rect",
    "from": {"data": "table"},
    "properties": {
      "enter": {
        "x": {"scale": "x", "field": "x"},
        "width": {"scale": "x", "band": true, "offset": -1},
        "y": {"scale": "y", "field": "y"},
        "y2": {"scale": "y", "value": 0},
        "fill": {"value": "steelblue"}
      }
    }
  }]
}
```



Data + Transforms

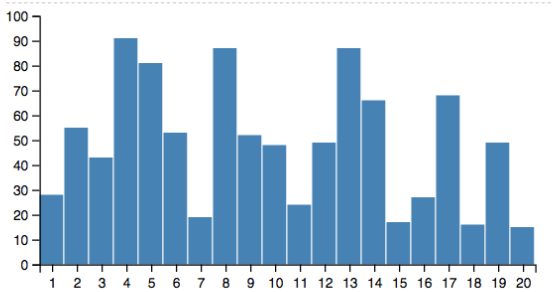
```
{
  "width": 400, "height": 200,
  "data": [
    {"name": "table", "url": "/data/sample.json"}
  ],
  "scales": [
    {
      "name": "x","type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "range": “height", "nice": true,
      "domain": {"data": "table", "field": "y"}
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [{
    "type": "rect",
    "from": {"data": "table"},
    "properties": {
      "enter": {
        "x": {"scale": "x", "field": "x"},
        "width": {"scale": "x", "band": true, "offset": -1},
        "y": {"scale": "y", "field": "y"},
        "y2": {"scale": "y", "value": 0},
        "fill": {"value": "steelblue"}
      }
    }
  }]
}
```



```
{
  "width": 400, "height": 200,
  "data": [
    {"name": "table", "url": "/data/sample.json"}
  ],
  "scales": [
    {
      "name": "x","type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "range": “height", "nice": true,
      "domain": {"data": "table", "field": "y"}
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [{
    "type": "rect",
    "from": {"data": "table"},
    "properties": {
      "enter": {
        "x": {"scale": "x", "field": "x"},
        "width": {"scale": "x", "band": true, "offset": -1},
        "y": {"scale": "y", "field": "y"},
        "y2": {"scale": "y", "value": 0},
        "fill": {"value": "steelblue"}
      }
    }
  }]
}
```

Data + Transforms

Scales

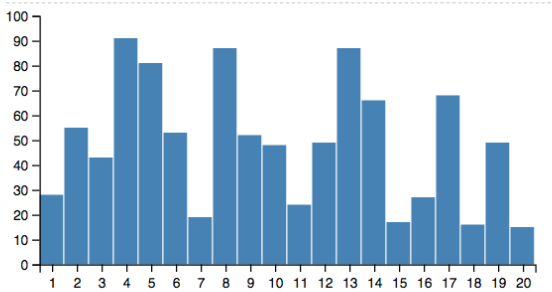


```
{
  "width": 400, "height": 200,
  "data": [
    {"name": "table", "url": "/data/sample.json"}
  ],
  "scales": [
    {
      "name": "x","type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "range": “height", "nice": true,
      "domain": {"data": "table", "field": "y"}
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [{
    "type": "rect",
    "from": {"data": "table"},
    "properties": {
      "enter": {
        "x": {"scale": "x", "field": "x"},
        "width": {"scale": "x", "band": true, "offset": -1},
        "y": {"scale": "y", "field": "y"},
        "y2": {"scale": "y", "value": 0},
        "fill": {"value": "steelblue"}
      }
    }
  }]
}
```

Data + Transforms

Scales

Guides



```
{
  "width": 400, "height": 200,
  "data": [
    {"name": "table", "url": "/data/sample.json"}
  ],
  "scales": [
    {
      "name": "x","type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "range": “height", "nice": true,
      "domain": {"data": "table", "field": "y"}
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [{
    "type": "rect",
    "from": {"data": "table"},
    "properties": {
      "enter": {
        "x": {"scale": "x", "field": "x"},
        "width": {"scale": "x", "band": true, "offset": -1},
        "y": {"scale": "y", "field": "y"},
        "y2": {"scale": "y", "value": 0},
        "fill": {"value": "steelblue"}
      }
    }
  }]
}
```

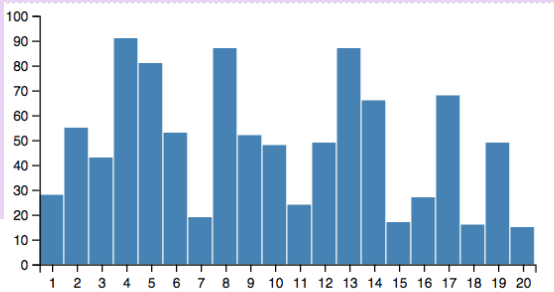
Data + Transforms

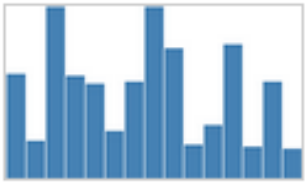
Scales

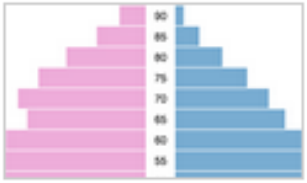
Guides

(Data + Transforms)

Marks







vega

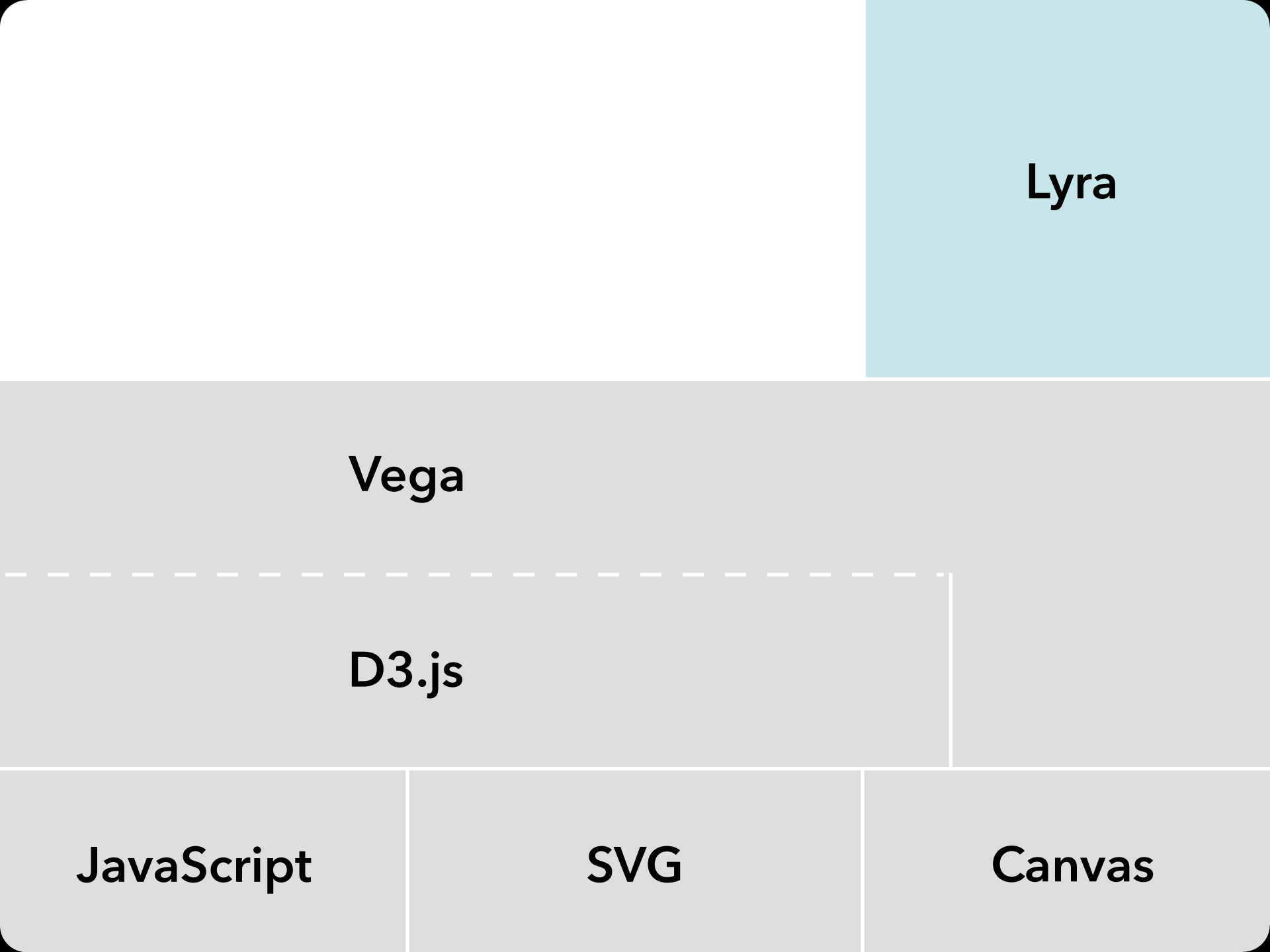
vega.min.js (188k)
Source (GitHub)

Vega is a visualization grammar, a declarative format for creating, saving and sharing visualization designs.

With Vega you can describe data visualizations in a JSON format, and generate interactive views using either HTML5 Canvas or SVG.

Read the tutorial, browse the documentation, join the discussion, and explore visualizations using the web-based Vega Editor.


Vega
D3.js
JavaScript
SVG
Canvas


Lyra
Vega
D3.js
JavaScript
SVG
Canvas

The Lyra Visualization Design Environment (VDE) alpha

Arvind Satyanarayan, Kanit "Ham" Wongsuphasawat, Jeffrey Heer

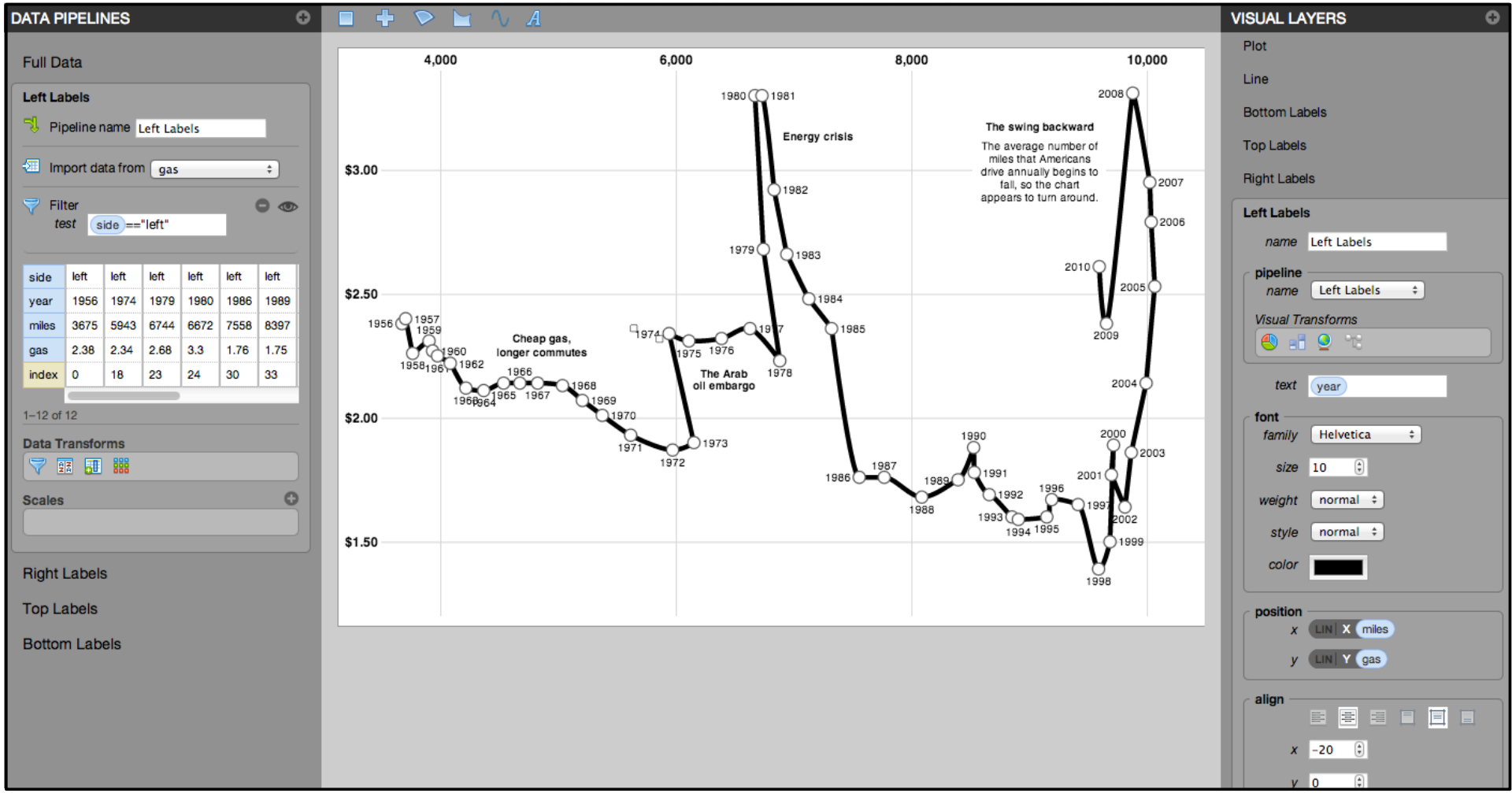
idl.cs.washington.edu/projects/lyra

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

ABSTRACT

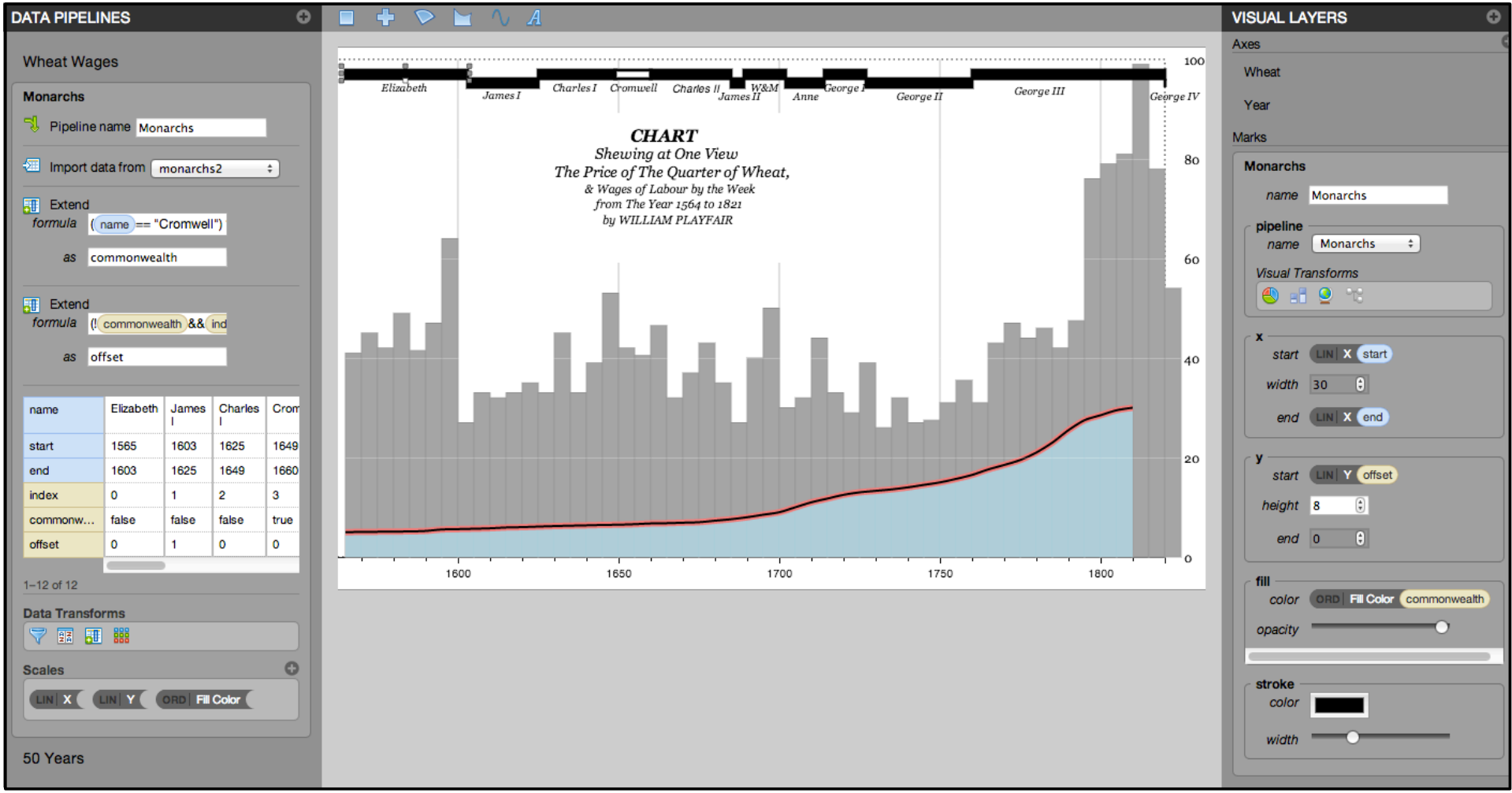
Lyra is an interactive environment that enables custom visualization design without writing any code. Graphical "marks" can be bound to data fields using property drop zones; dynamically positioned using connectors; and directly moved, rotated, and resized using handles. Lyra also provides a data pipeline interface for iterative visual specification of data transformations and layout algorithms. Lyra is more expressive than interactive systems like Tableau, allowing designers to create custom visualizations comparable to hand-coded visualizations built with D3 or Processing. These visualizations can then be easily published and reused on the Web.

Lyra A Visualization Design Environment



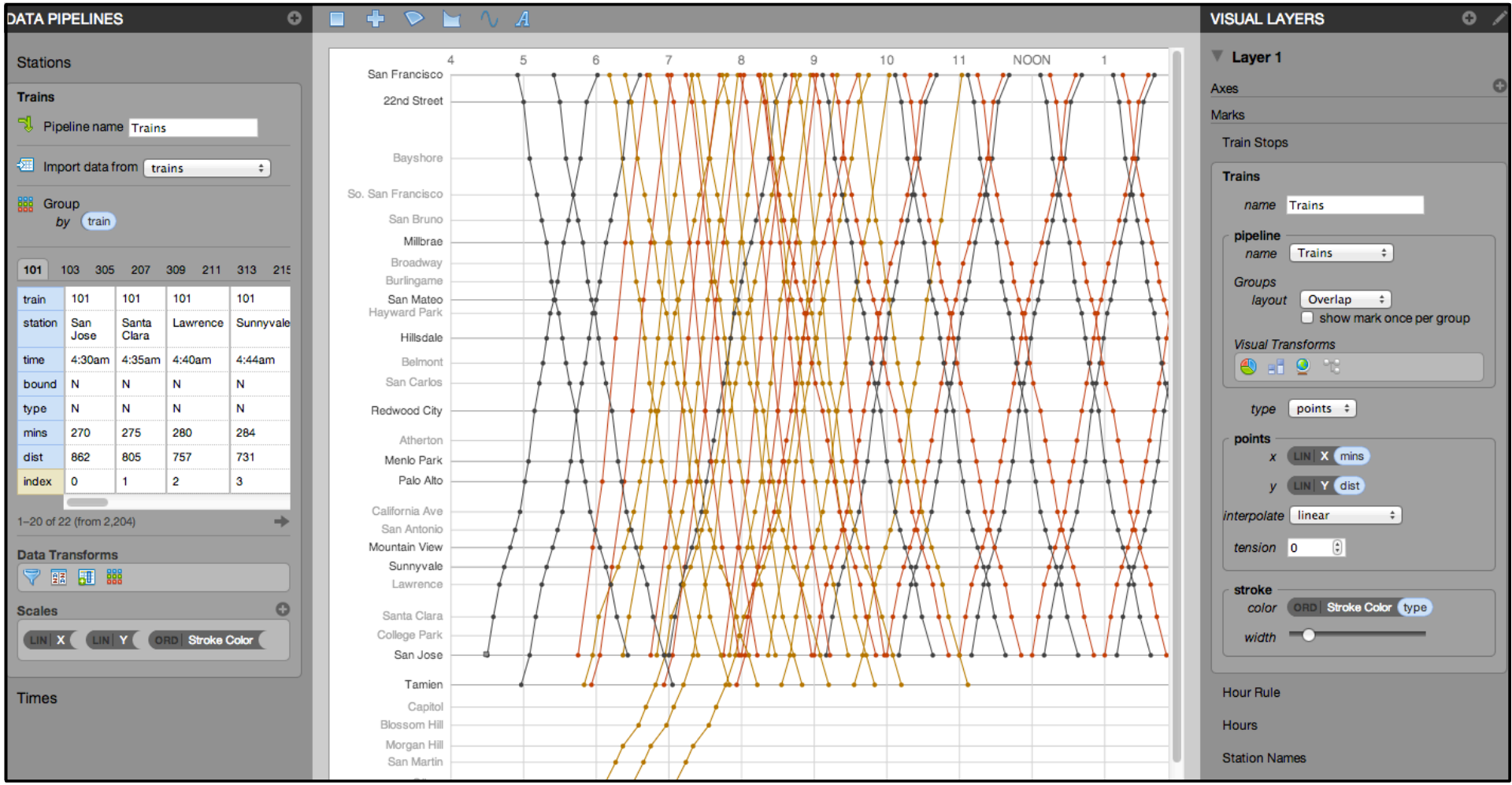
Driving Shifts into Reverse by Hannah Fairfield, NYTimes

Lyra A Visualization Design Environment

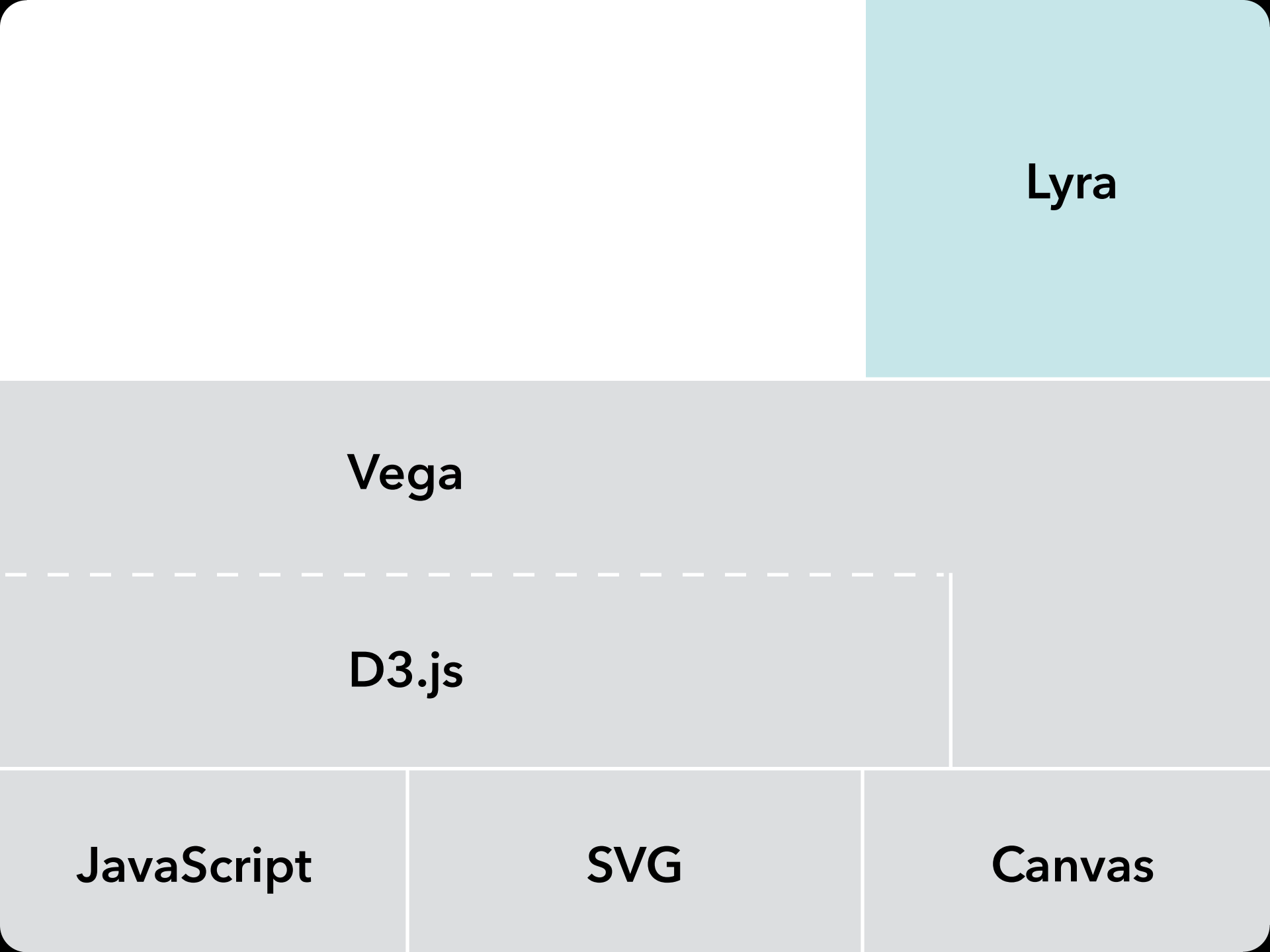


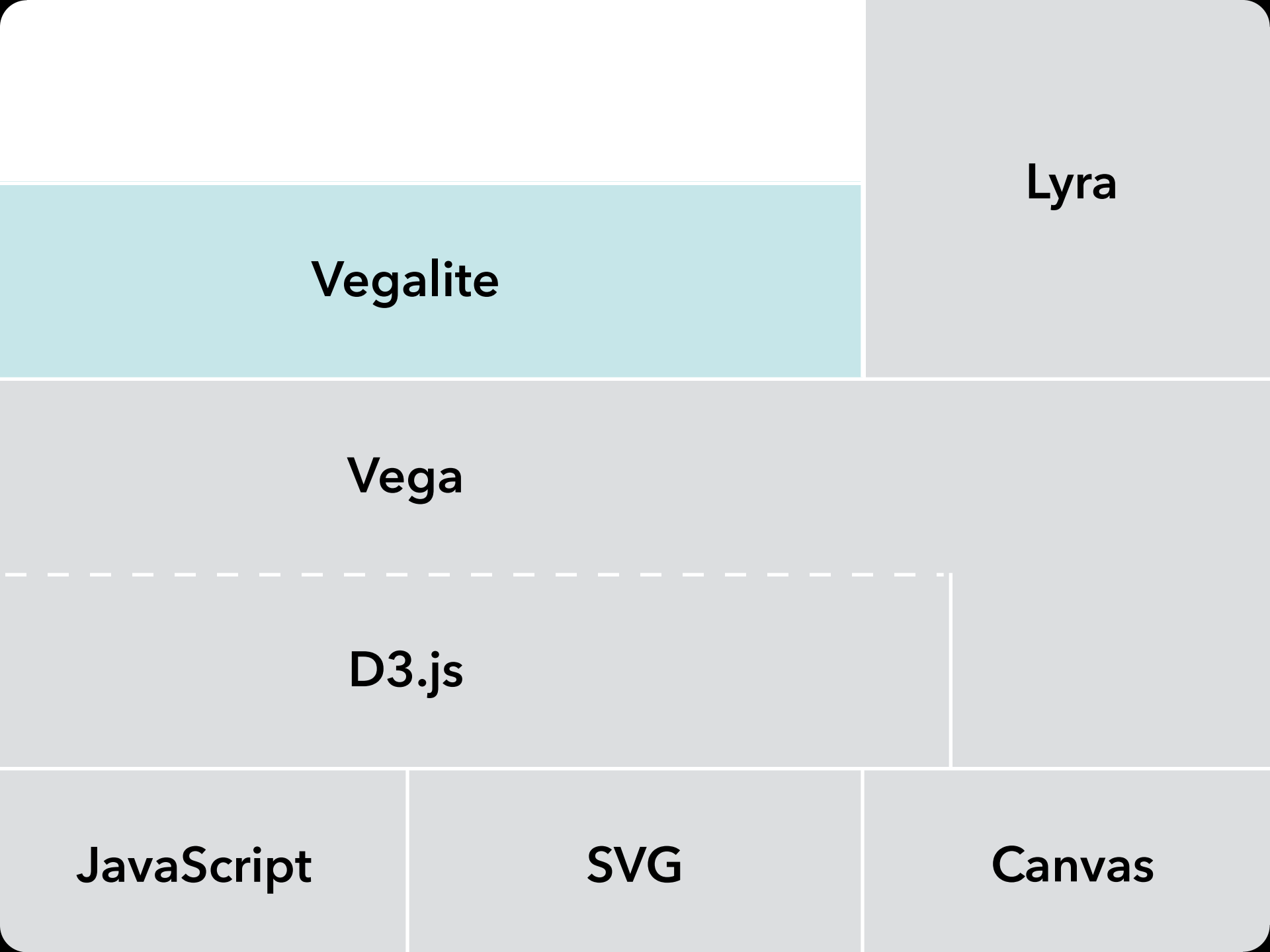
by William Playfair

Lyra A Visualization Design Environment



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


Lyra
Vega
D3.js
JavaScript
SVG
Canvas


Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas

Vegalite

A formal model for statistical graphics
Inspired by *Grammar of Graphics* & *Tableau*

Includes **data transformation** & **encoding**

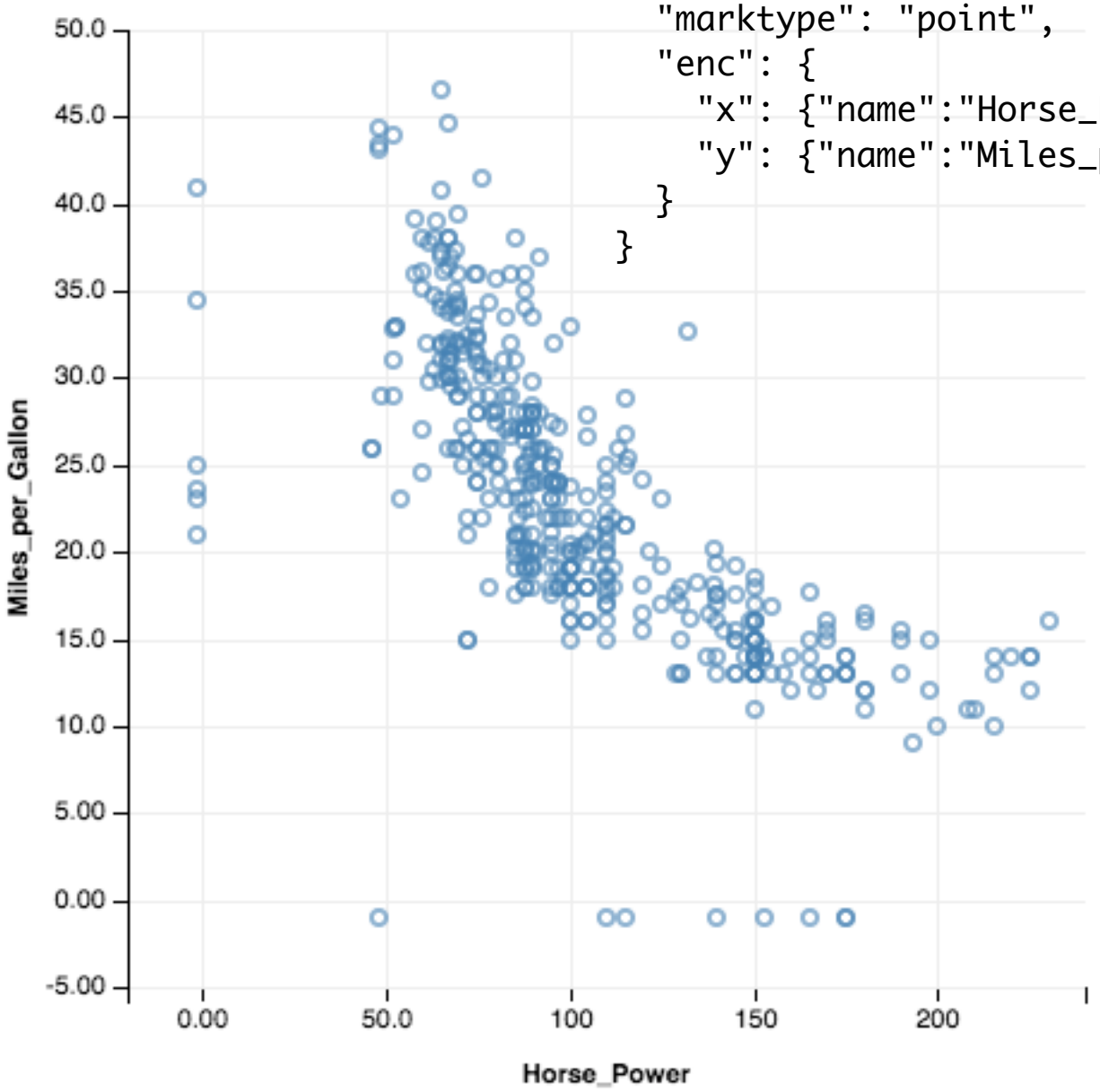
Vegalite

A formal model for statistical graphics
Inspired by *Grammar of Graphics* & *Tableau*

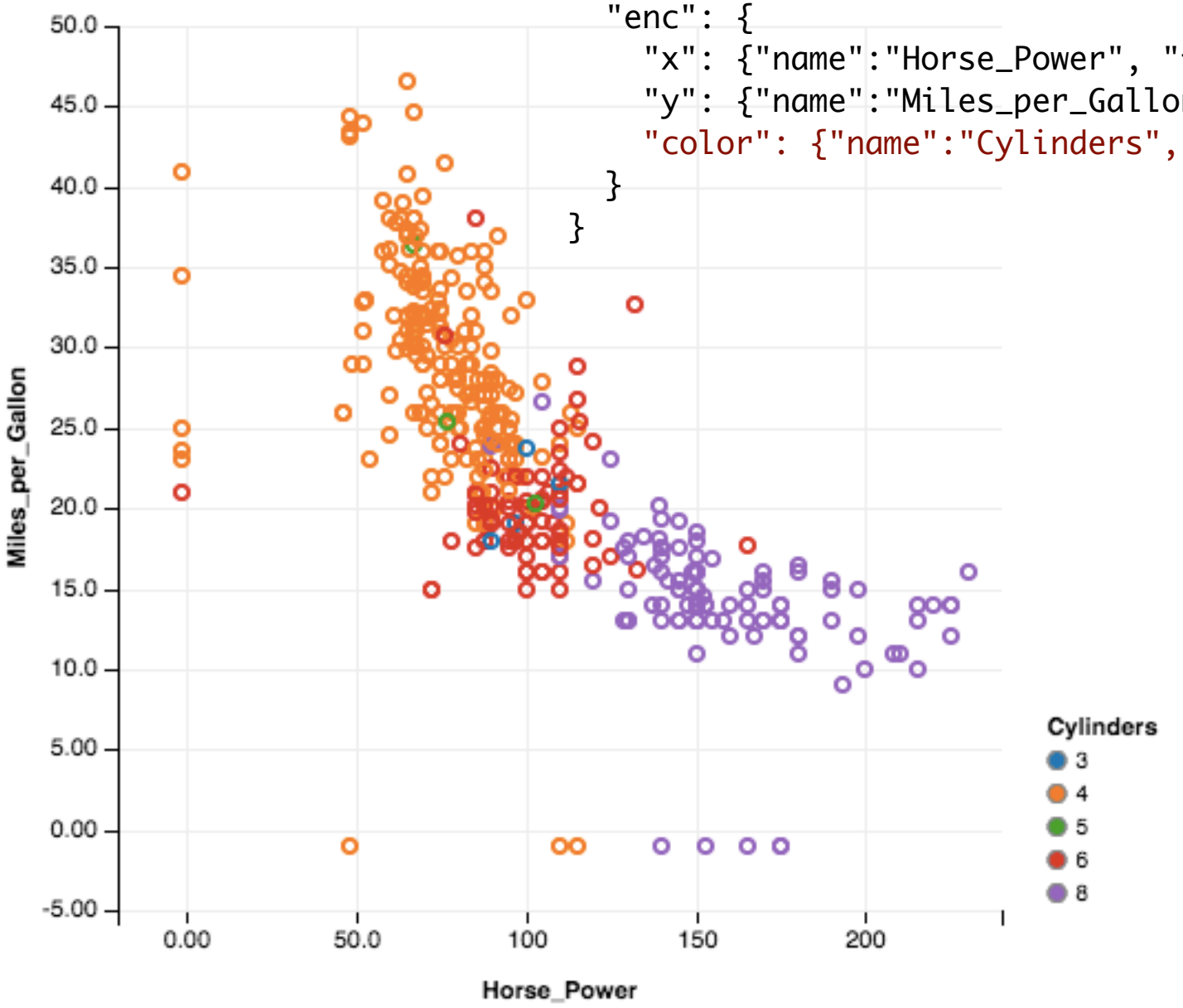
Includes **data transformation** & **encoding**

Uses a simple, concise **JSON format** that compiles to full-blown **Vega specifications**

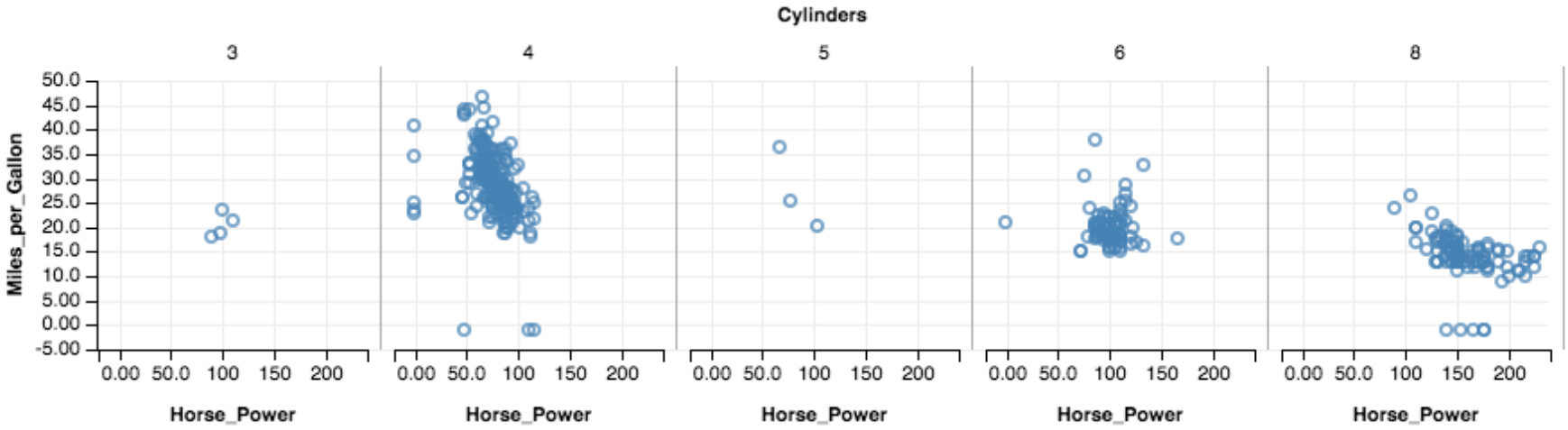
Easy **programmatic generation**!



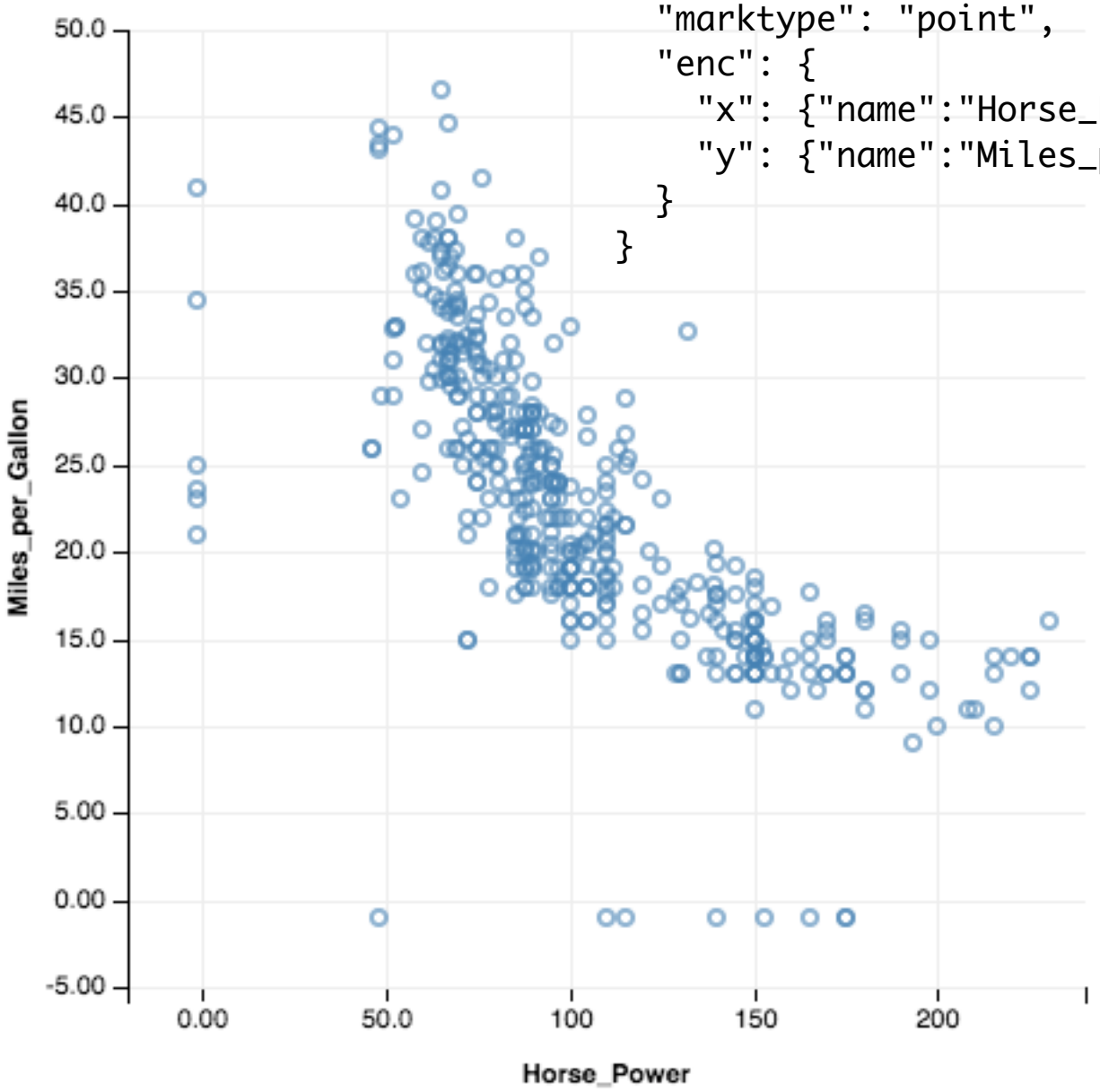

```
{
  "marktype": "point",
  "enc": {
    "x": {"name":"Horse_Power", "type":"Q"},
    "y": {"name":"Miles_per_Gallon", "type":"Q"}
  }
}
```



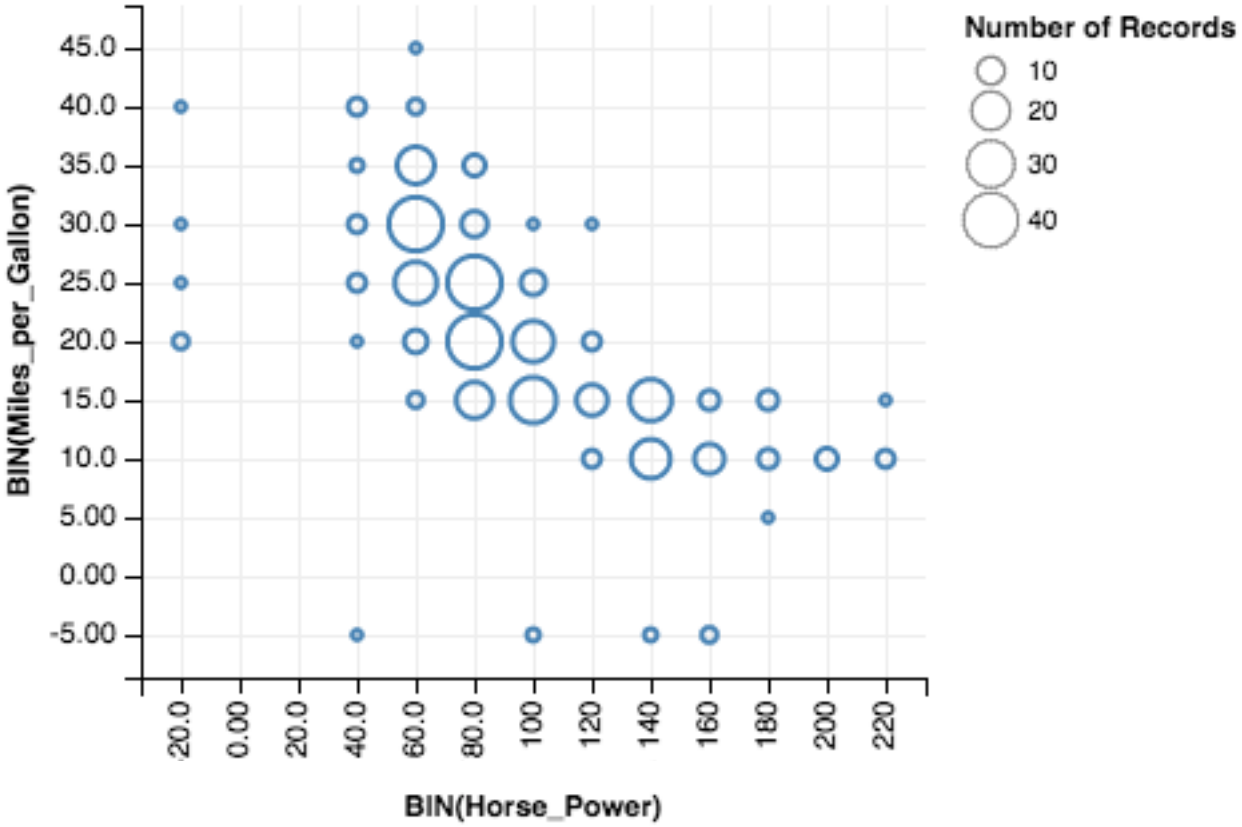

```
{
  "marktype": "point",
  "enc": {
    "x": {"name":"Horse_Power", "type":"Q"},
    "y": {"name":"Miles_per_Gallon", "type":"Q"},
    "color": {"name":"Cylinders", "type":"O"}
  }
}
```



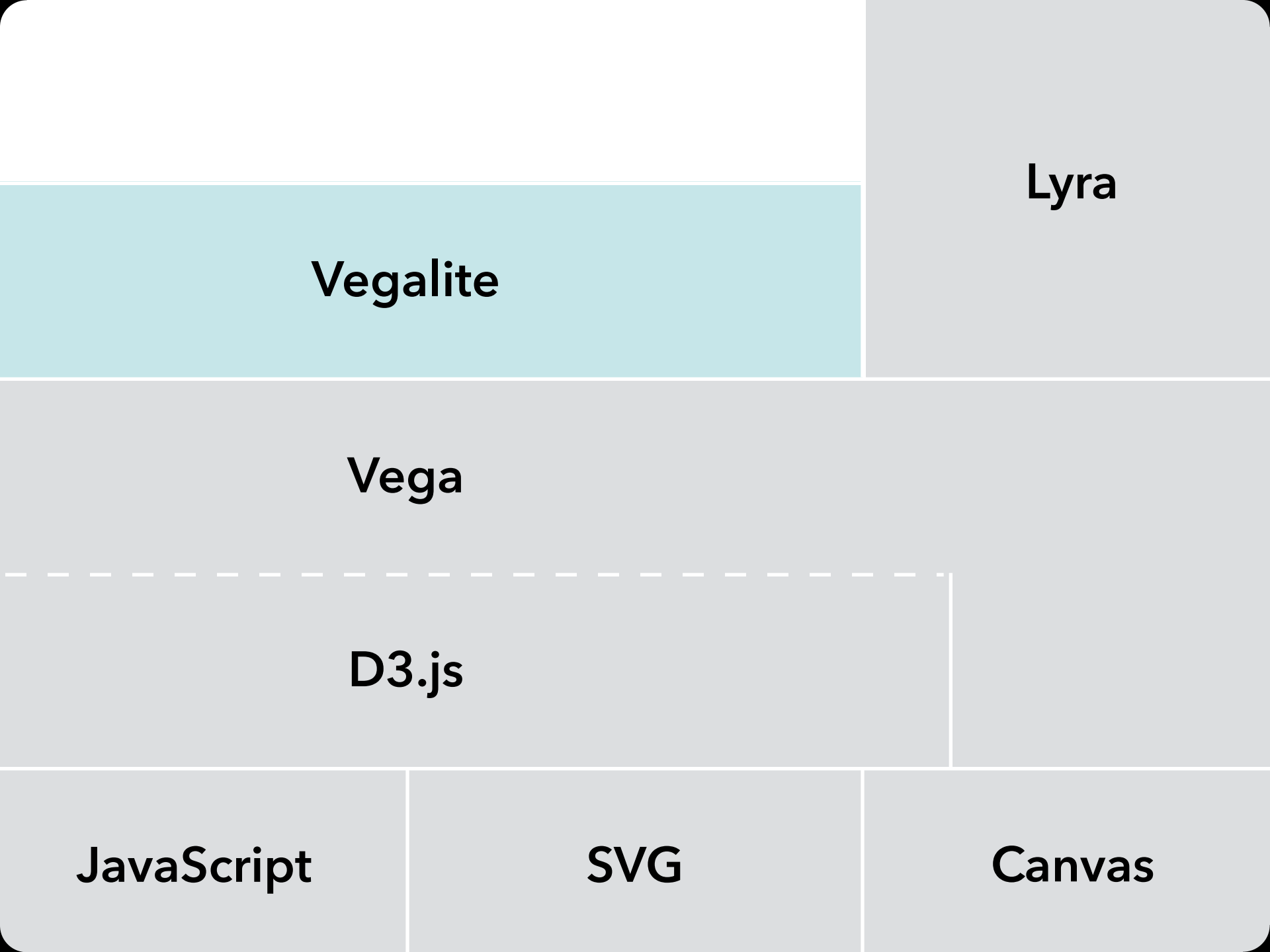

```
{
  "marktype": "point",
  "enc": {
    "x": {"name":"Horse_Power", "type":"Q"},
    "y": {"name":"Miles_per_Gallon", "type":"Q"},
    "col": {"name":"Cylinders", "type":"O"}
  }
}
```

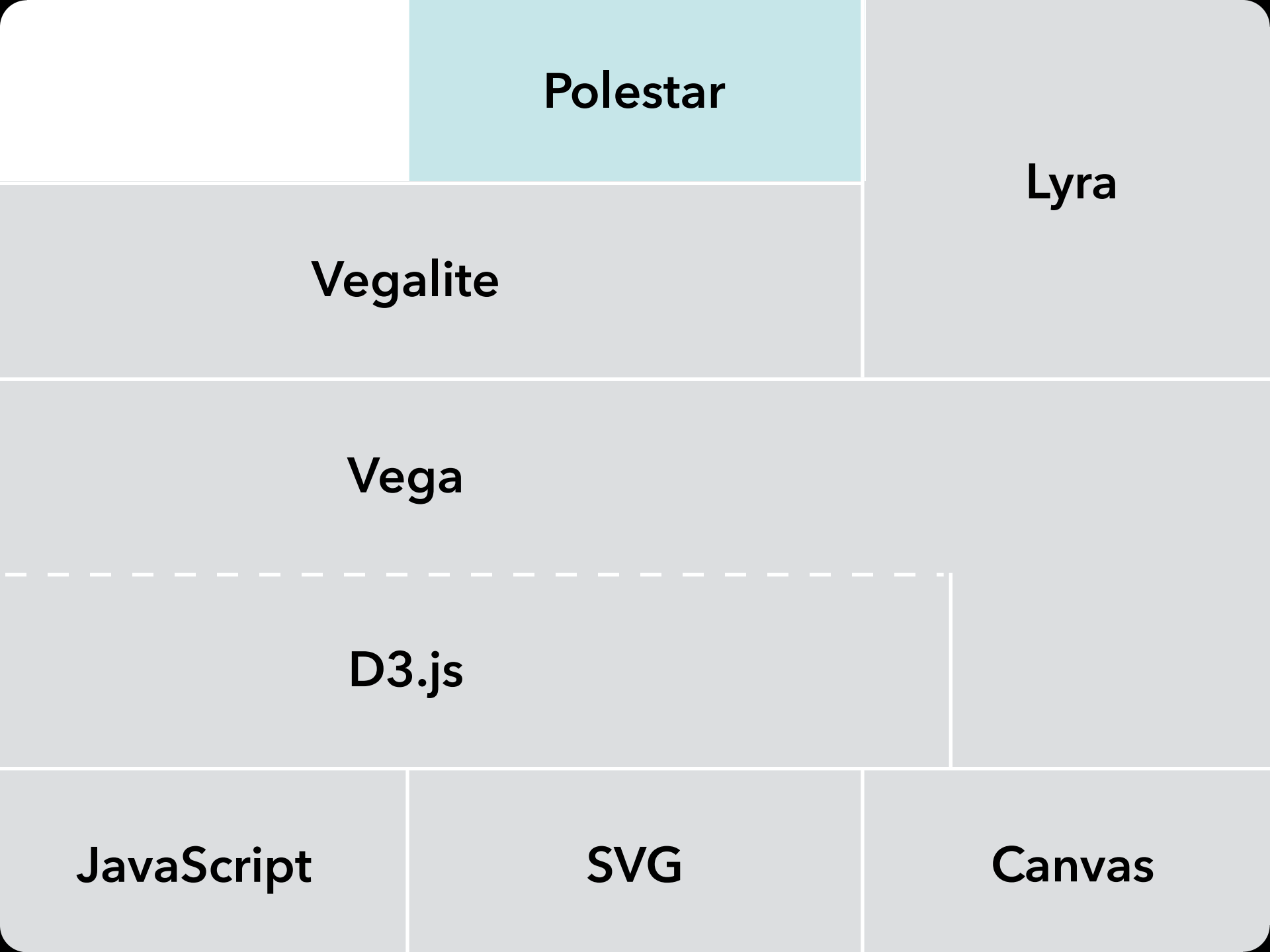



```
{
  "marktype": "point",
  "enc": {
    "x": {"name":"Horse_Power", "type":"Q"},
    "y": {"name":"Miles_per_Gallon", "type":"Q"}
  }
}
```




```
{
  "marktype": "point",
  "enc": {
    "x": {"name":"Horse_Power", "type":"Q", "bin":{"maxbins":15}},
    "y": {"name":"Miles_per_Gallon", "type":"Q", "bin":{"maxbins":15}},
    "size": {"name":"*", "type":"Q", "aggr":"count"}
  }
}
```


Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas


Polestar
Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas

Polestar

A graphical interface for **Vegalite**

Rapid visualization via drag-and-drop

Named in honor of **Polaris**, the research project that led to **Tableau**.

Pole★

https://uwdata.github.io/polestar

Pole★

Bookmarks (0) Undo Redo

DATA

Cars

- Abc Cylinders
- Abc name
- Abc Origin
- Year
- # Acceleration
- # Displacement
- # Horse Power
- # Miles per Gallon
- # Weight in lbs
- # COUNT

ENCODING

Clear

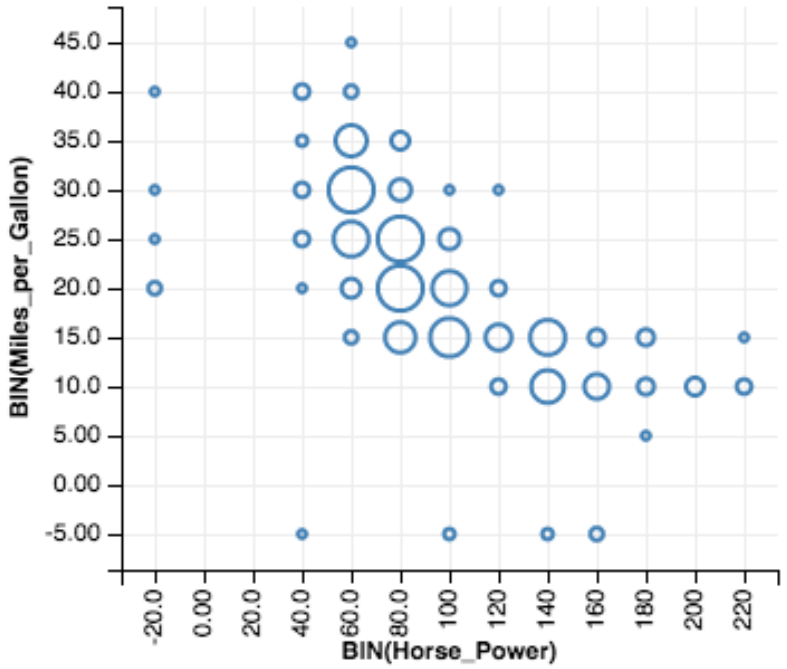
Positional

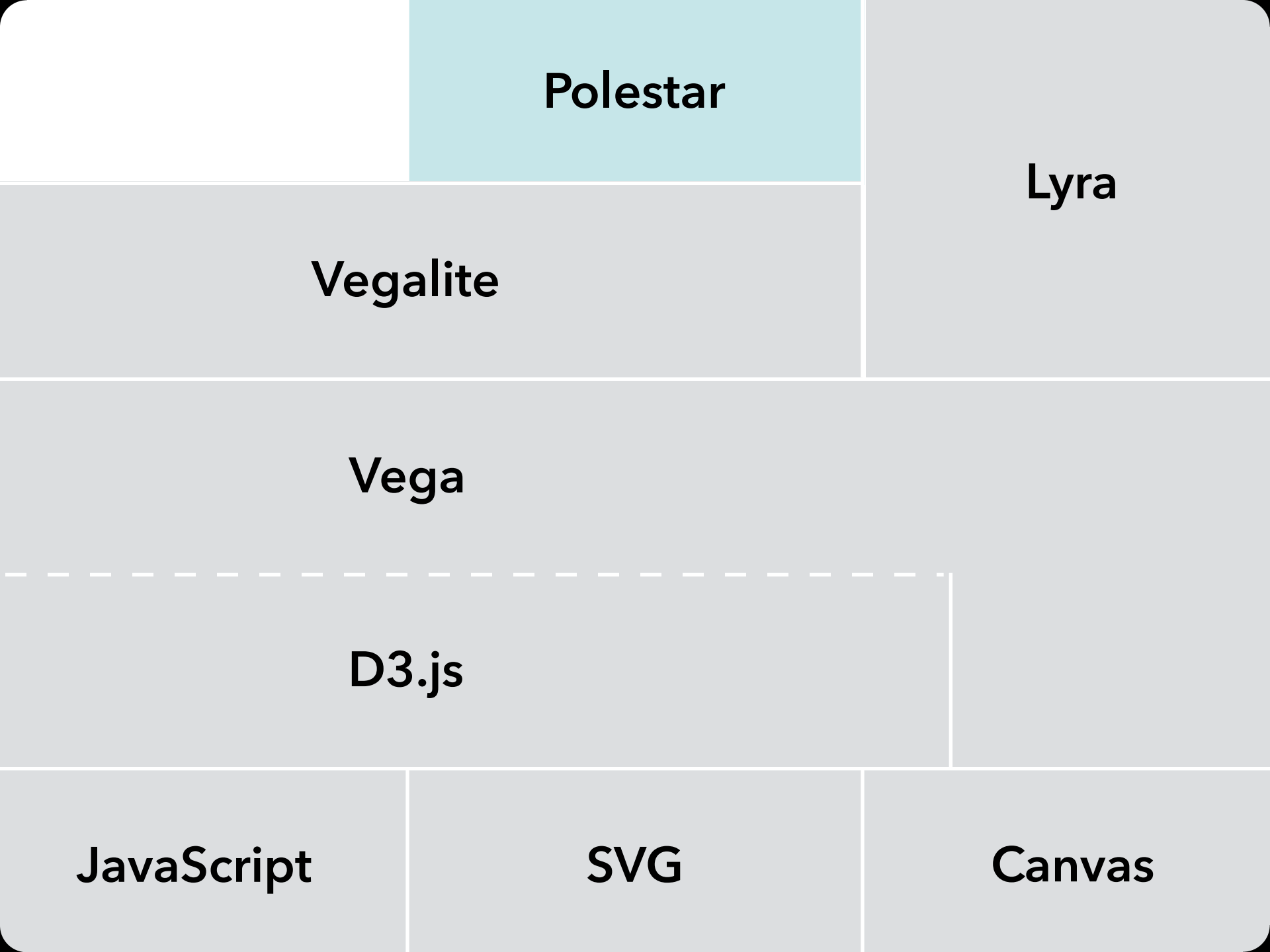
- x: # BIN (Horse
- y: # BIN (Miles per
- col: drop a field here
- row: drop a field here

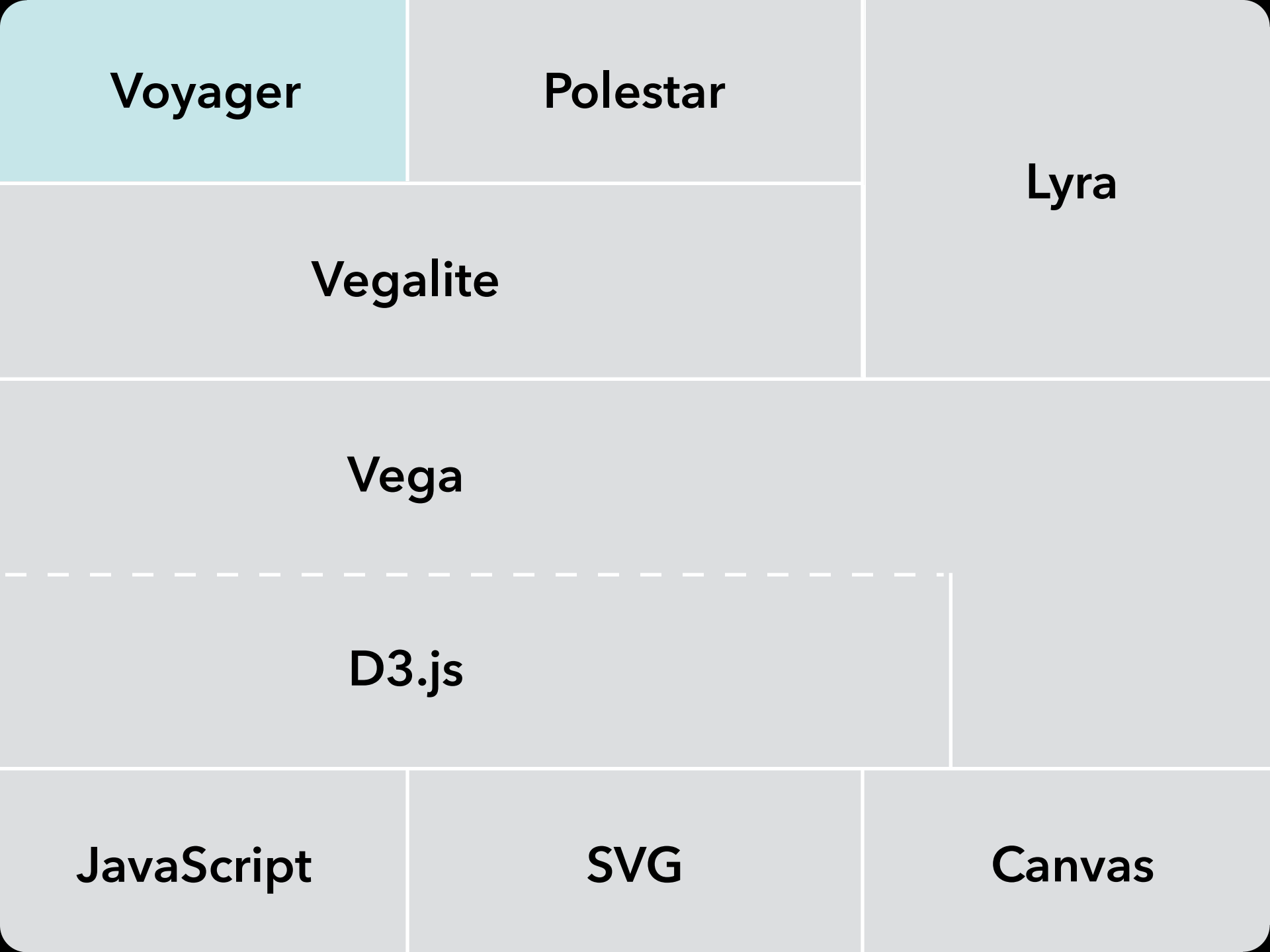
Marks

point

- size: # COUNT
- color: drop a field here
- shape: drop a field here
- detail: drop a field here
- text: drop a field here




Polestar
Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas


Voyager
Polestar
Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas

Voyager

Reduce tedious manual specification

Voyager

Reduce tedious manual specification

Support early-stage data exploration

Encourage *data coverage*

Discourage *premature fixation*

Voyager

Reduce tedious manual specification

Support early-stage data exploration
Encourage *data coverage*
Discourage *premature fixation*

Approach: browse a gallery of visualizations

Voyager

Reduce tedious manual specification

Support early-stage data exploration
Encourage *data coverage*
Discourage *premature fixation*

Approach: browse a gallery of visualizations
Challenge - *combinatorial explosion!*

Voyager

Reduce tedious manual specification

Support early-stage data exploration
Encourage *data coverage*
Discourage *premature fixation*

Approach: browse a gallery of visualizations
Challenge - *combinatorial explosion!*
Automatic recommendation of useful views
+ **end-user steering** to focus exploration

Data Voyager

https://uwdata.github.io/voyager

Data Voyager

Bookmarks (0) Undo Redo

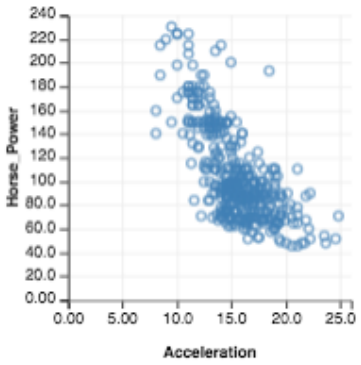
DATA Cars

- Cylinders
- name
- Origin
- AUTO (Year)
- AUTO (Acceleration)
- AUTO (Displacement)
- AUTO (Horse Power)
- AUTO (Miles per Gallon)
- AUTO (Weight in lbs)
- COUNT

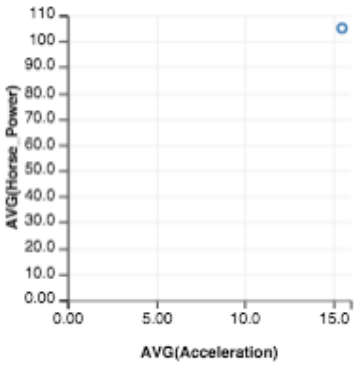
Reset

Showing Data Variations for # AUTO (Acceleration) # AUTO (Horse Power)

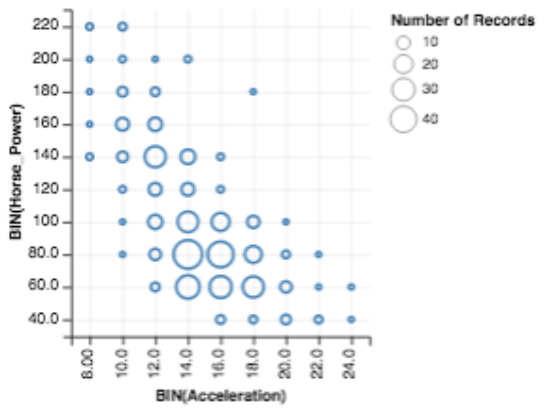
Acceleration # Horse Power



AVG (Acceleration) # AVG (Horse Power)

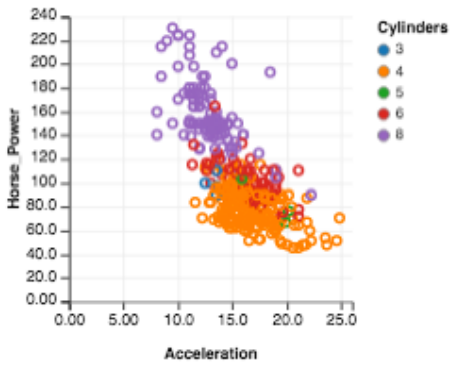


BIN (Acceleration) # BIN (Horse Power) # COUNT

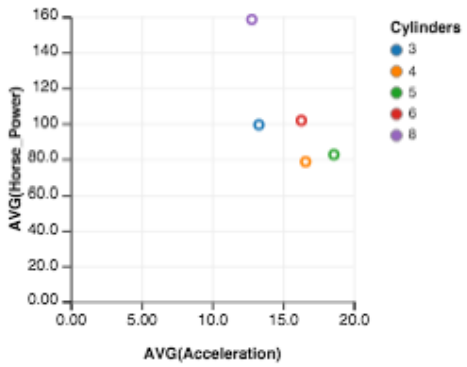


Showing Data Variations for # AUTO (Acceleration) # AUTO (Horse Power) **with one additional field.**

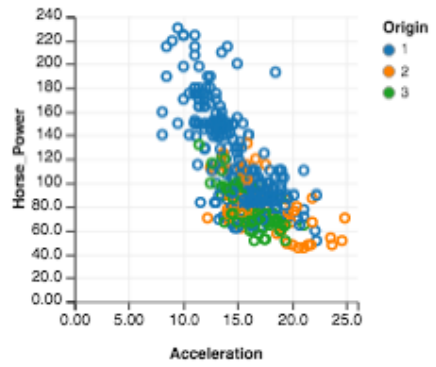
Acceleration # Horse Power Abc Cylinders



AVG (Acceleration) # AVG (Horse Power) Abc Cylinders



Acceleration # Horse Power Abc Origin



AVG (Acceleration) # AVG (Horse Power) Abc Origin

Acceleration # Horse Power YEAR (Year)

User


Data Set
Voyager
Visualization Browser
User

User

Compass

Recommendation Engine

Data Schema
& Statistics

Voyager

Visualization Browser

1. Select **data variables**
2. Apply **transformations**
3. Pick visual **encodings**

User

Compass
Recommendation Engine

Data Schema
& Statistics

Voyager
Visualization Browser

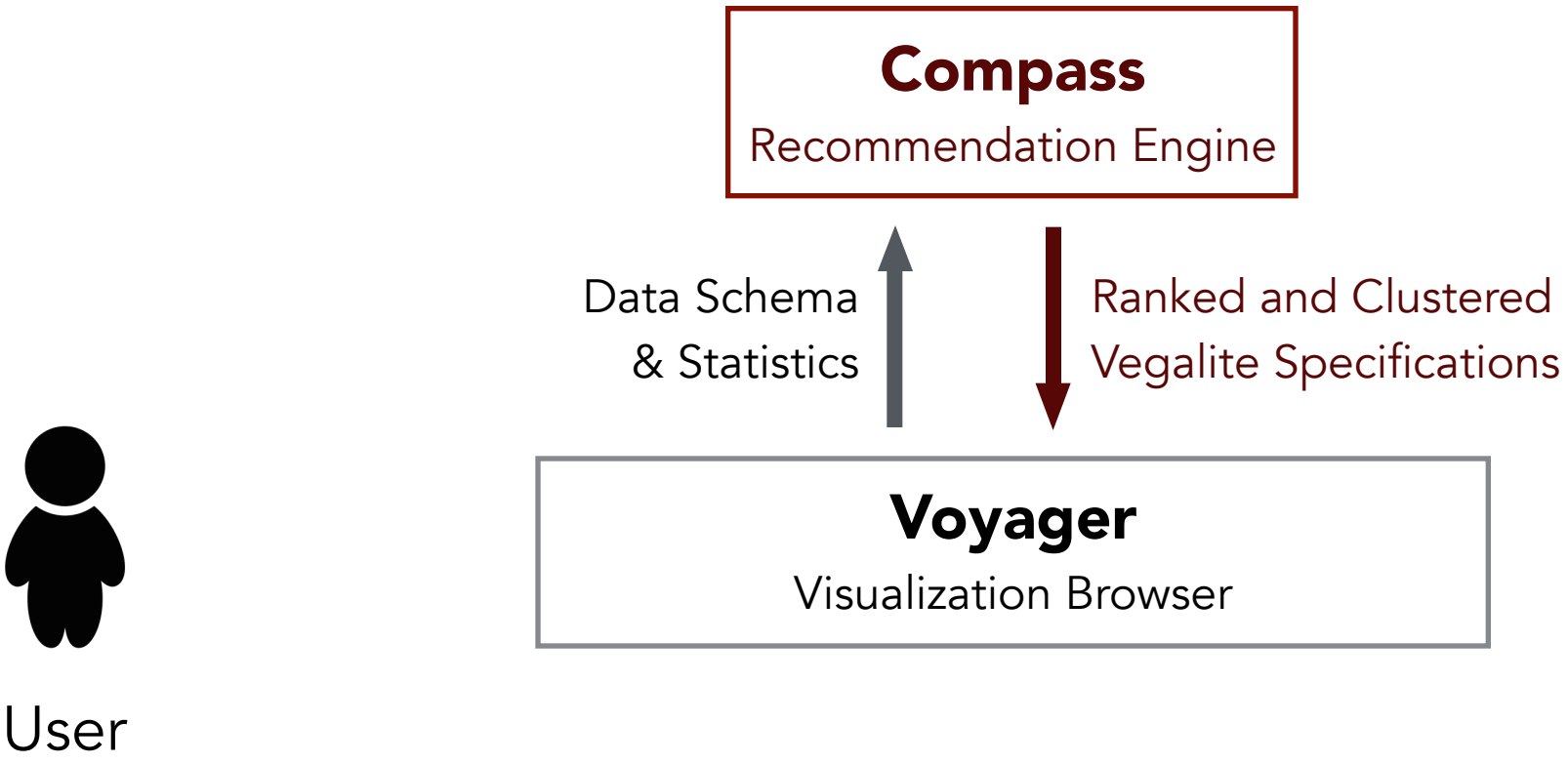
Constrain & rank choices by **data type**, **statistics** & **perceptual principles.**

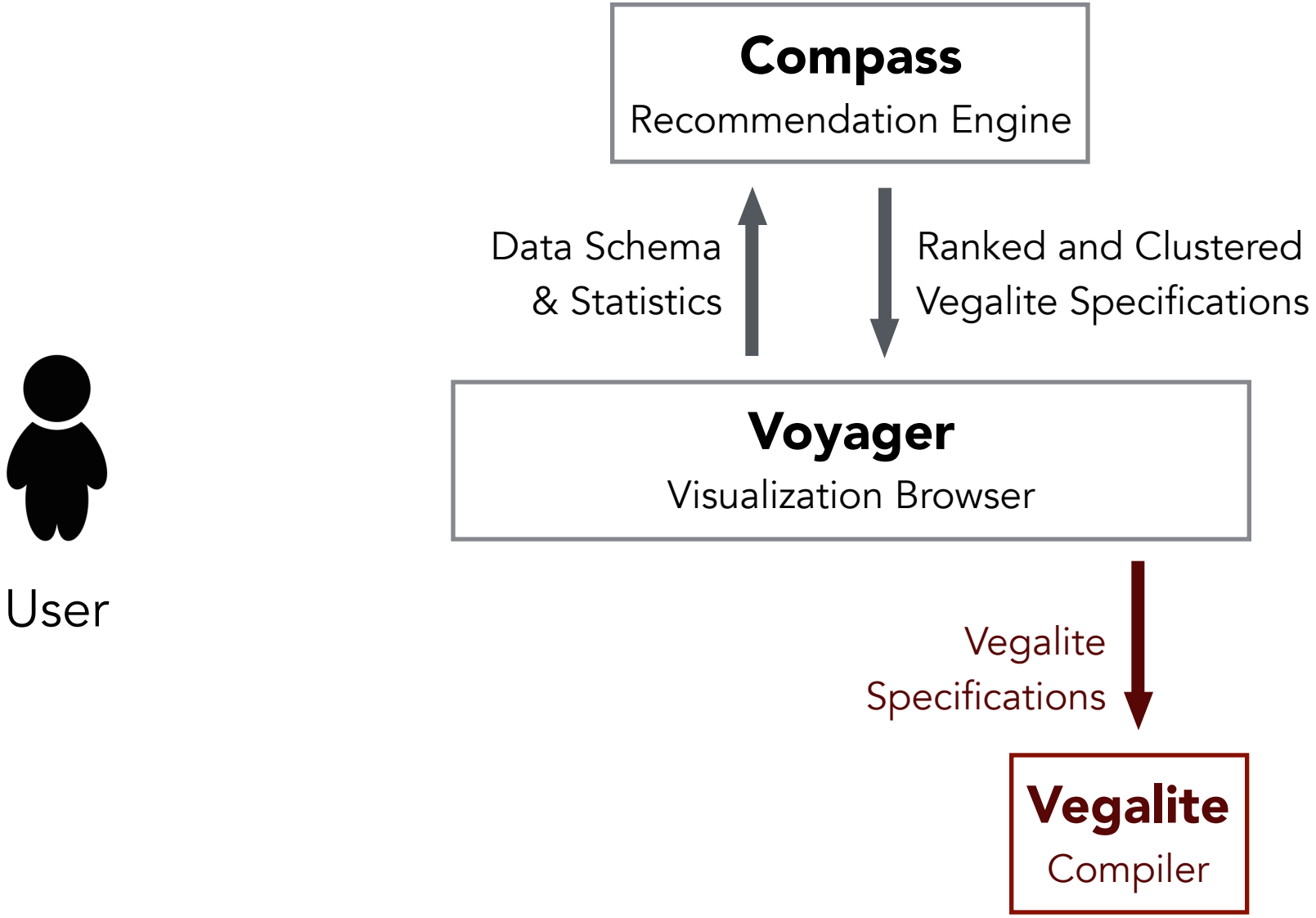
User

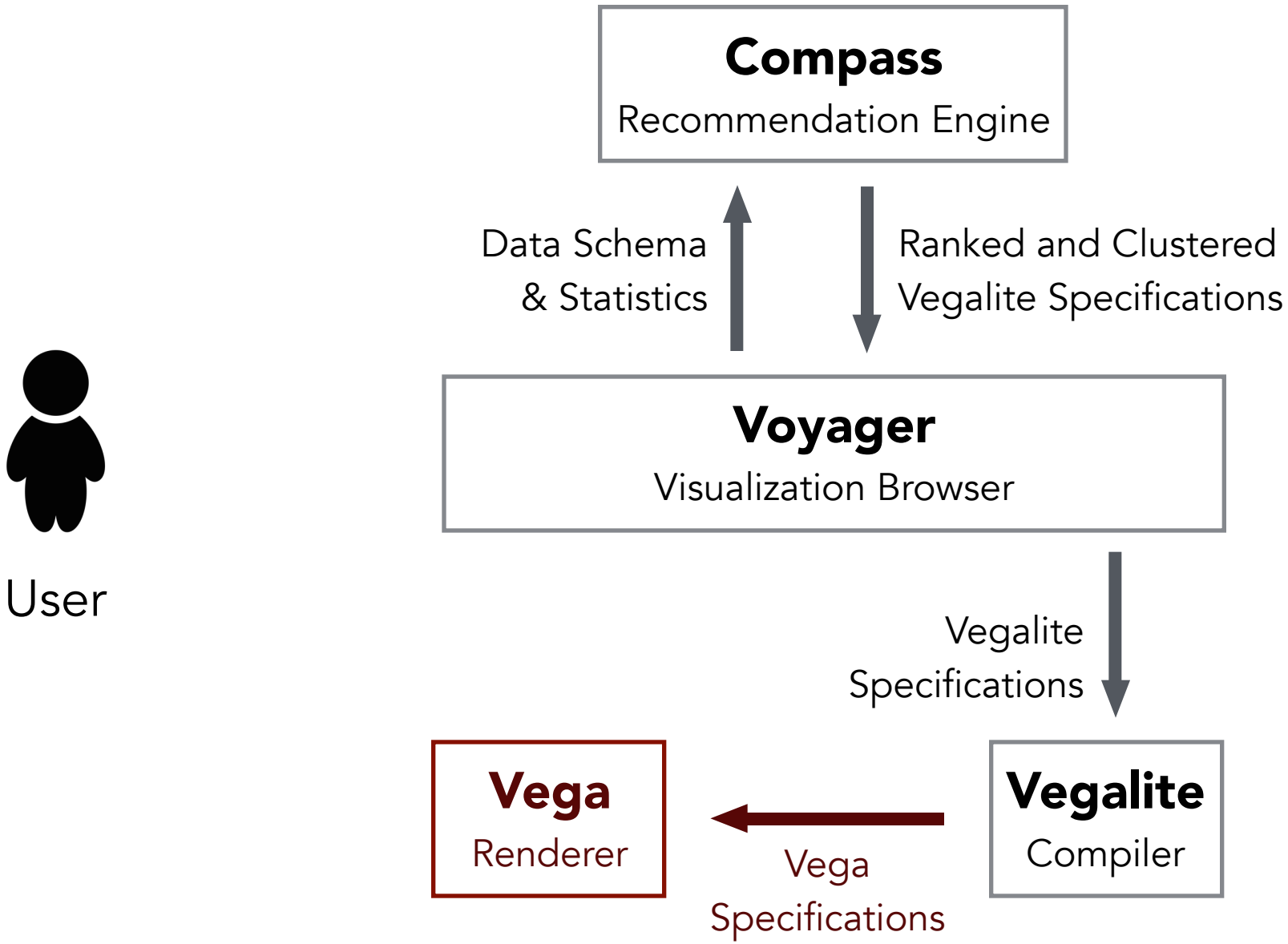
Compass
Recommendation Engine

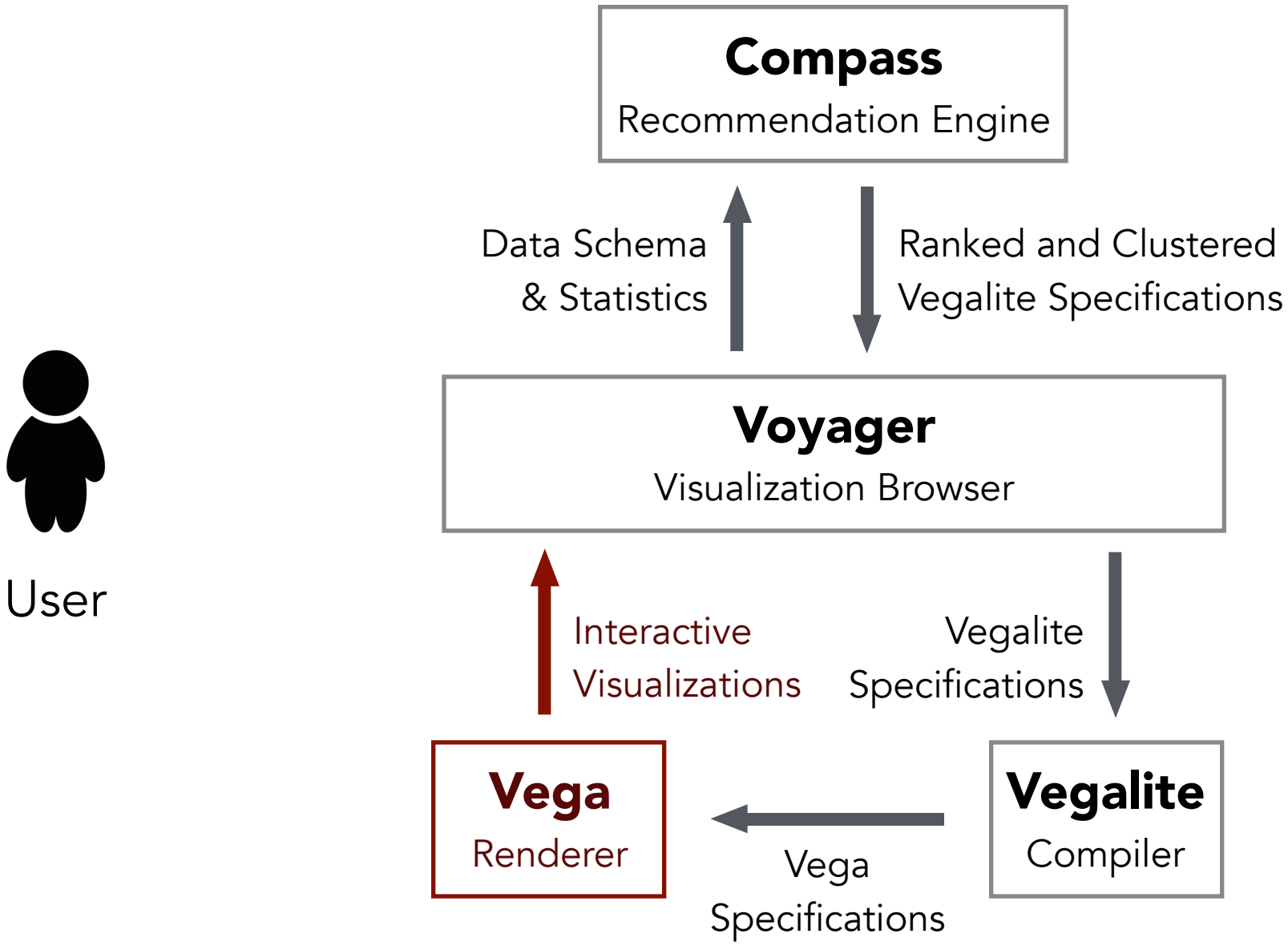
Data Schema
& Statistics

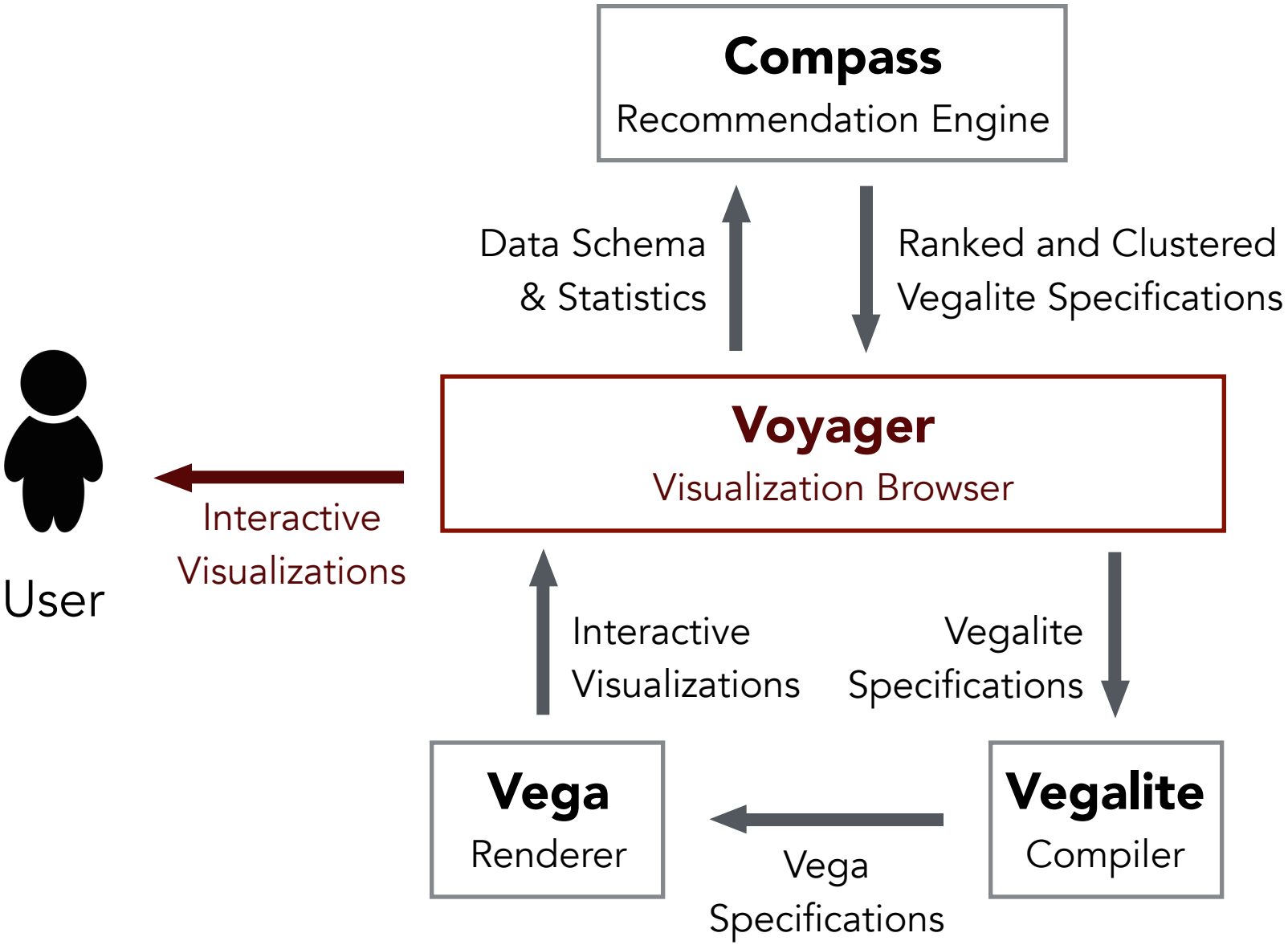
Voyager
Visualization Browser

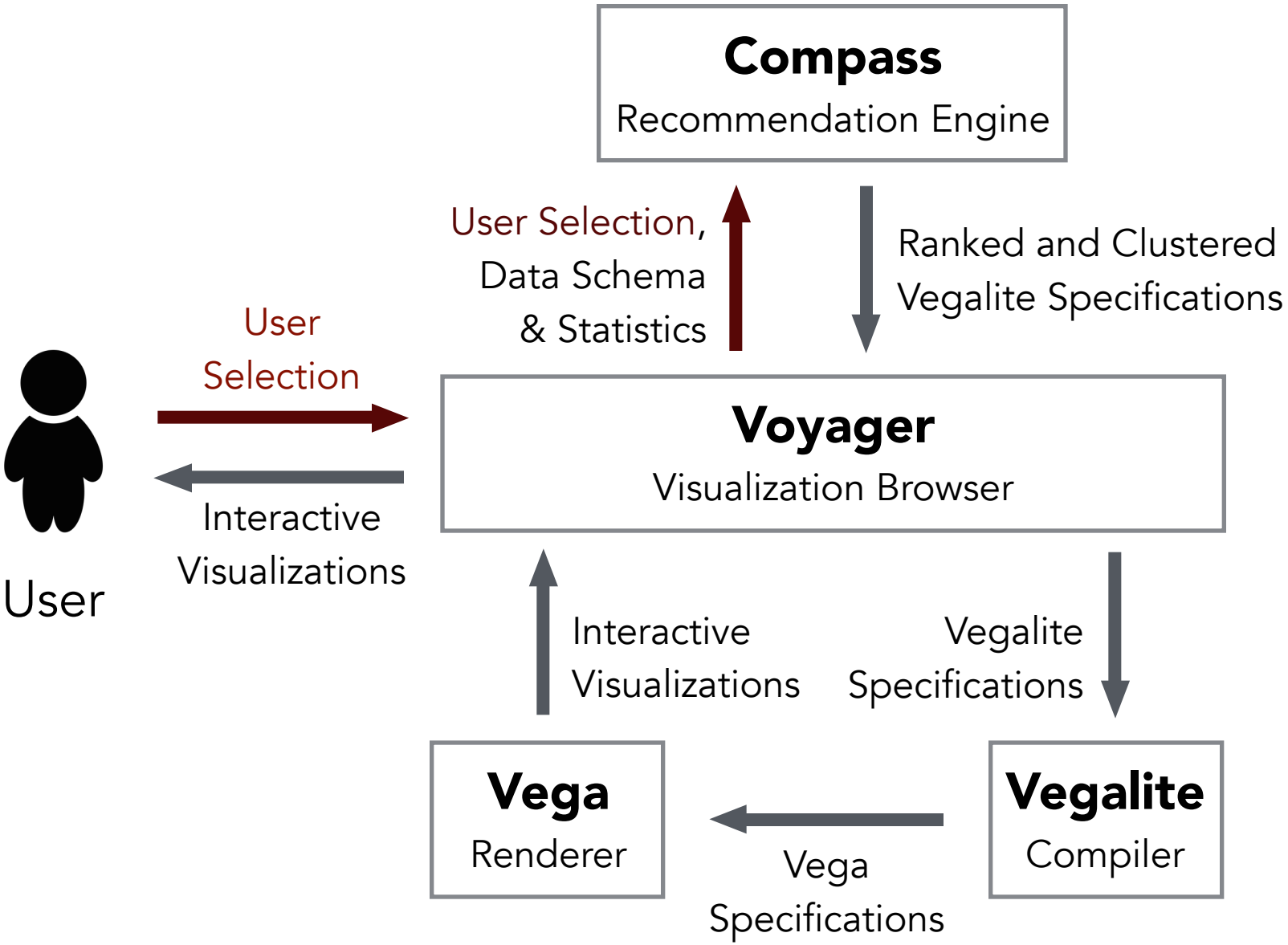

Compass
Recommendation Engine
Data Schema
& Statistics
Ranked and Clustered
Vegalite Specifications
Voyager
Visualization Browser
User


Compass
Recommendation Engine
Data Schema
& Statistics
Ranked and Clustered
Vegalite Specifications
Voyager
Visualization Browser
User
Vegalite
Specifications
Vegalite
Compiler


Compass
Recommendation Engine
Data Schema
& Statistics
Ranked and Clustered
Vegalite Specifications
Voyager
Visualization Browser
User
Vegalite
Specifications
Vega
Renderer
Vega
Specifications
Vegalite
Compiler


Compass
Recommendation Engine
Data Schema
& Statistics
Ranked and Clustered
Vegalite Specifications
Voyager
Visualization Browser
User
Interactive
Visualizations
Vegalite
Specifications
Vega
Renderer
Vega
Specifications
Vegalite
Compiler

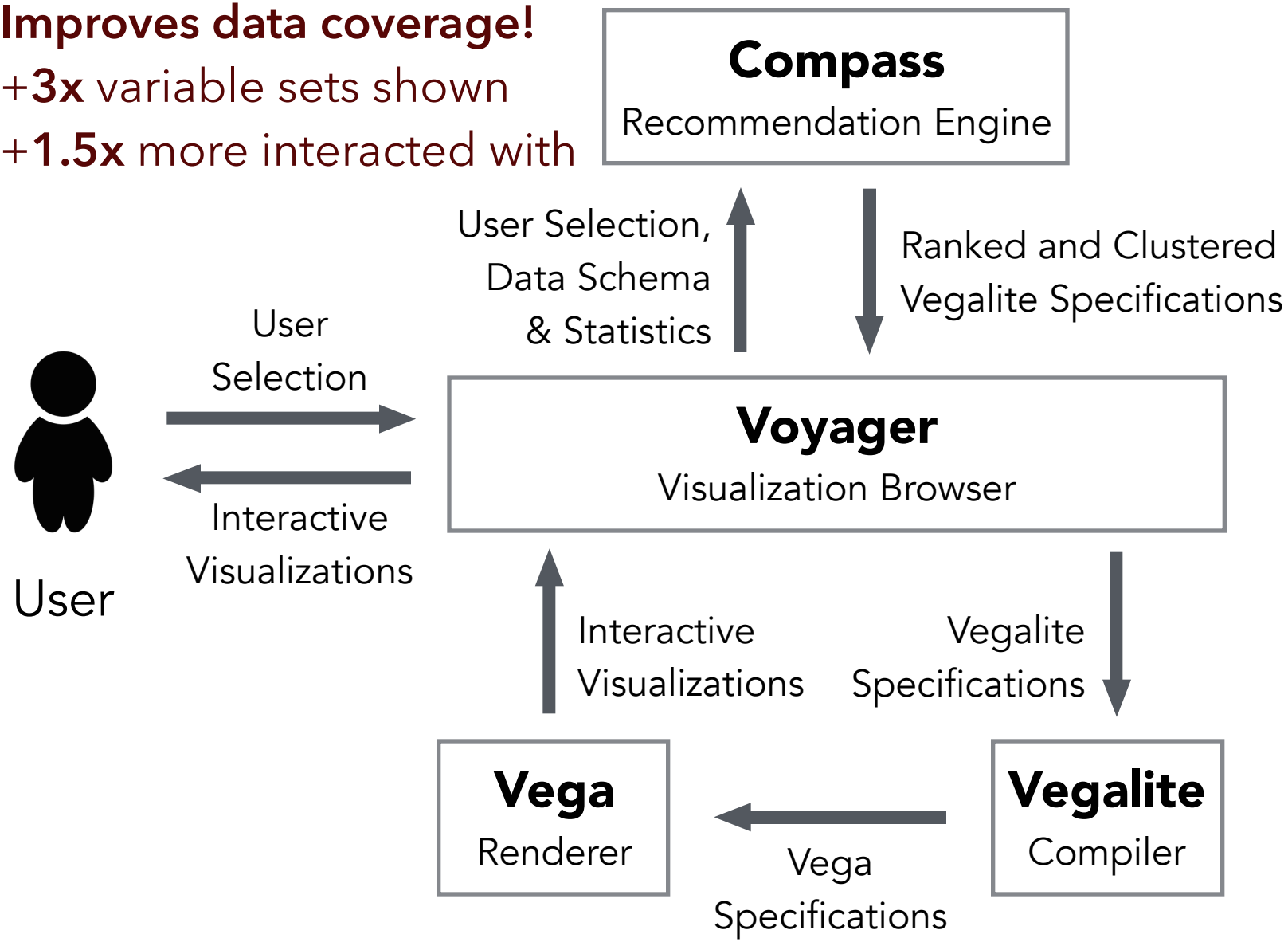

Compass
Recommendation Engine
Data Schema
& Statistics
Ranked and Clustered
Vegalite Specifications
Voyager
Visualization Browser
User
Interactive
Visualizations
Interactive
Visualizations
Vegalite
Specifications
Vega
Renderer
Vegalite
Compiler
Vega
Specifications

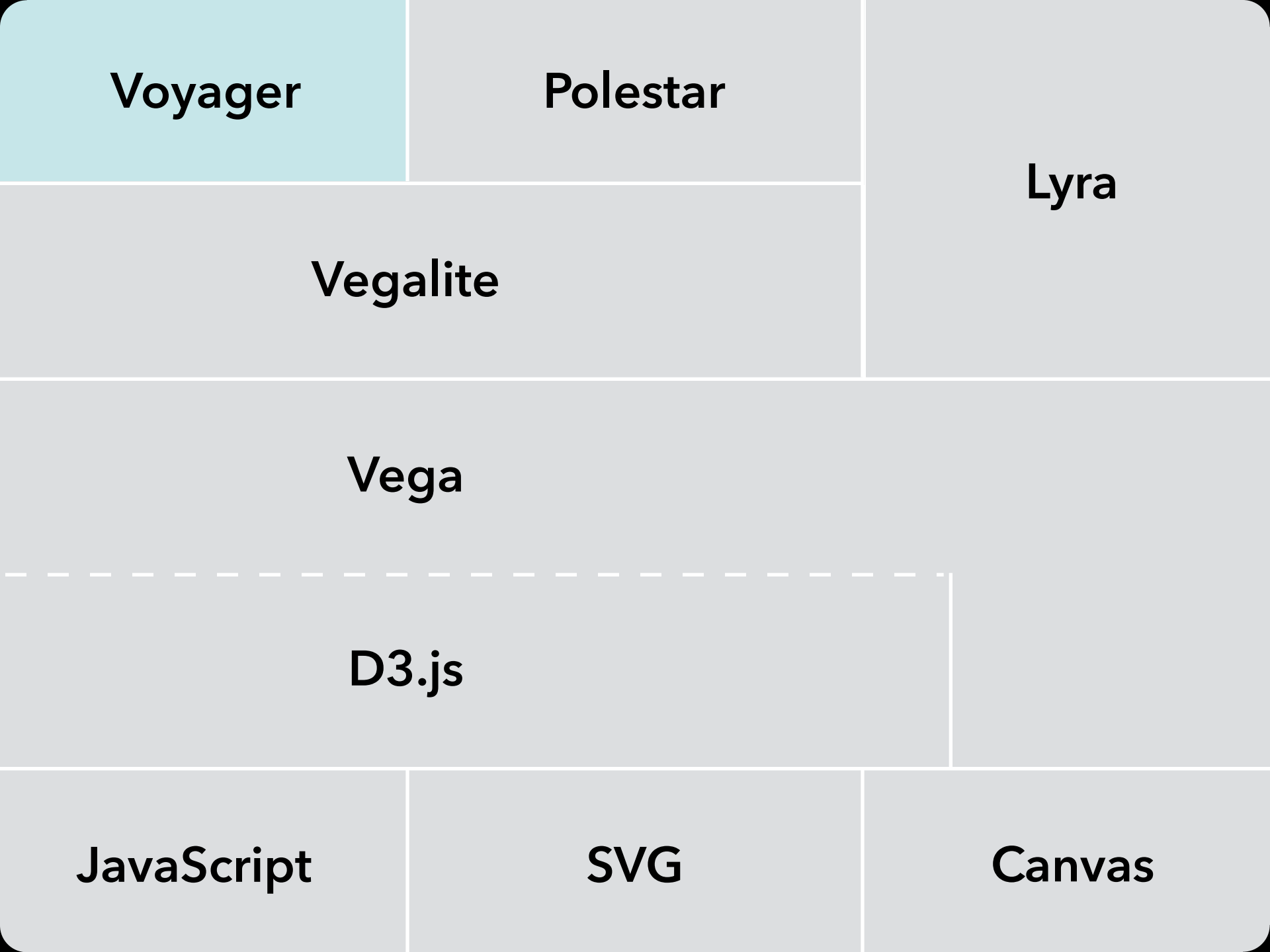

Compass
Recommendation Engine
User Selection,
Data Schema
& Statistics
Ranked and Clustered
Vegalite Specifications
User
Selection
Interactive
Visualizations
Voyager
Visualization Browser
User
Interactive
Visualizations
Vegalite
Specifications
Vega
Renderer
Vega
Specifications
Vegalite
Compiler

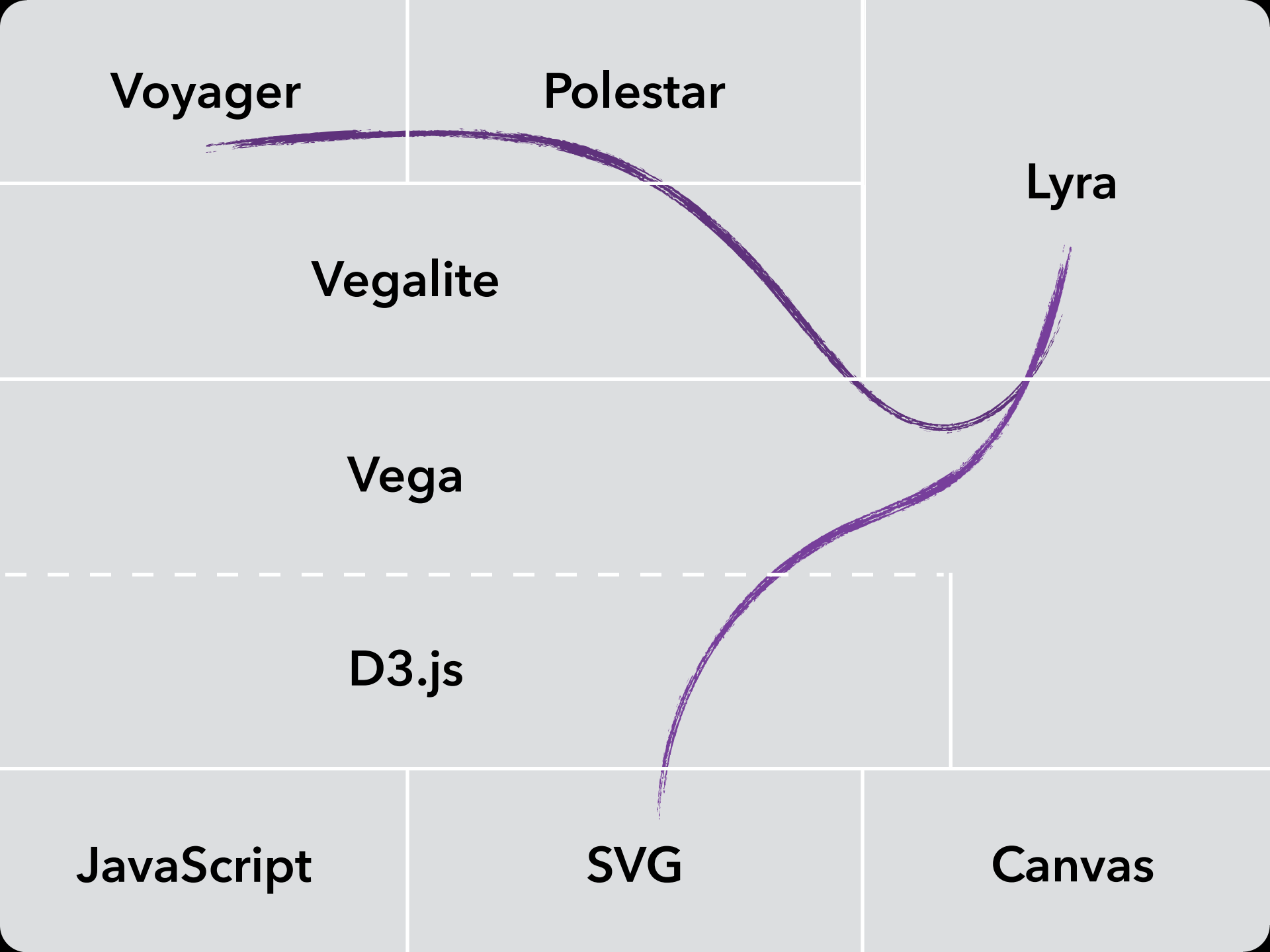
Improves data coverage!

+**3x** variable sets shown

+**1.5x** more interacted with

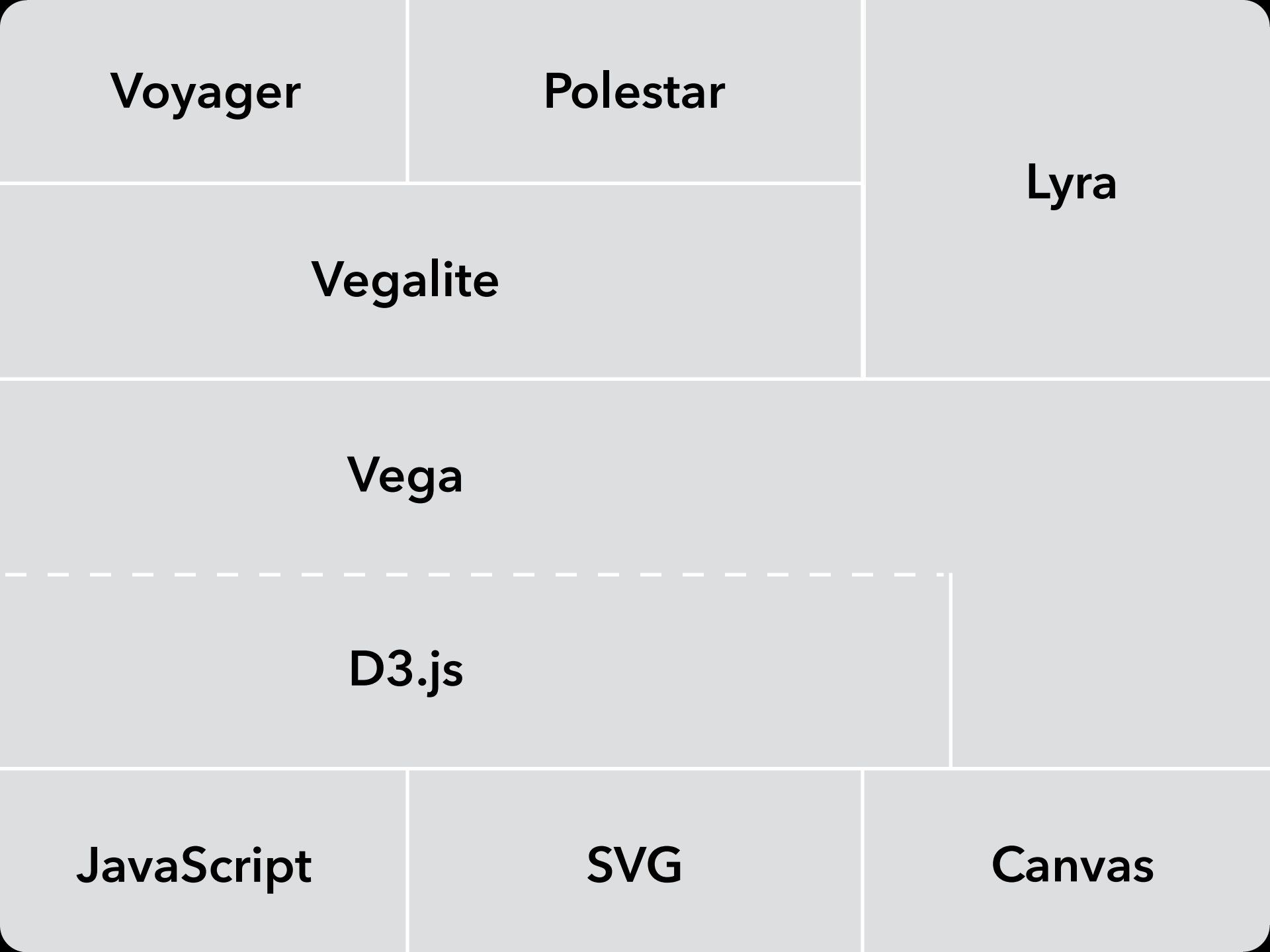


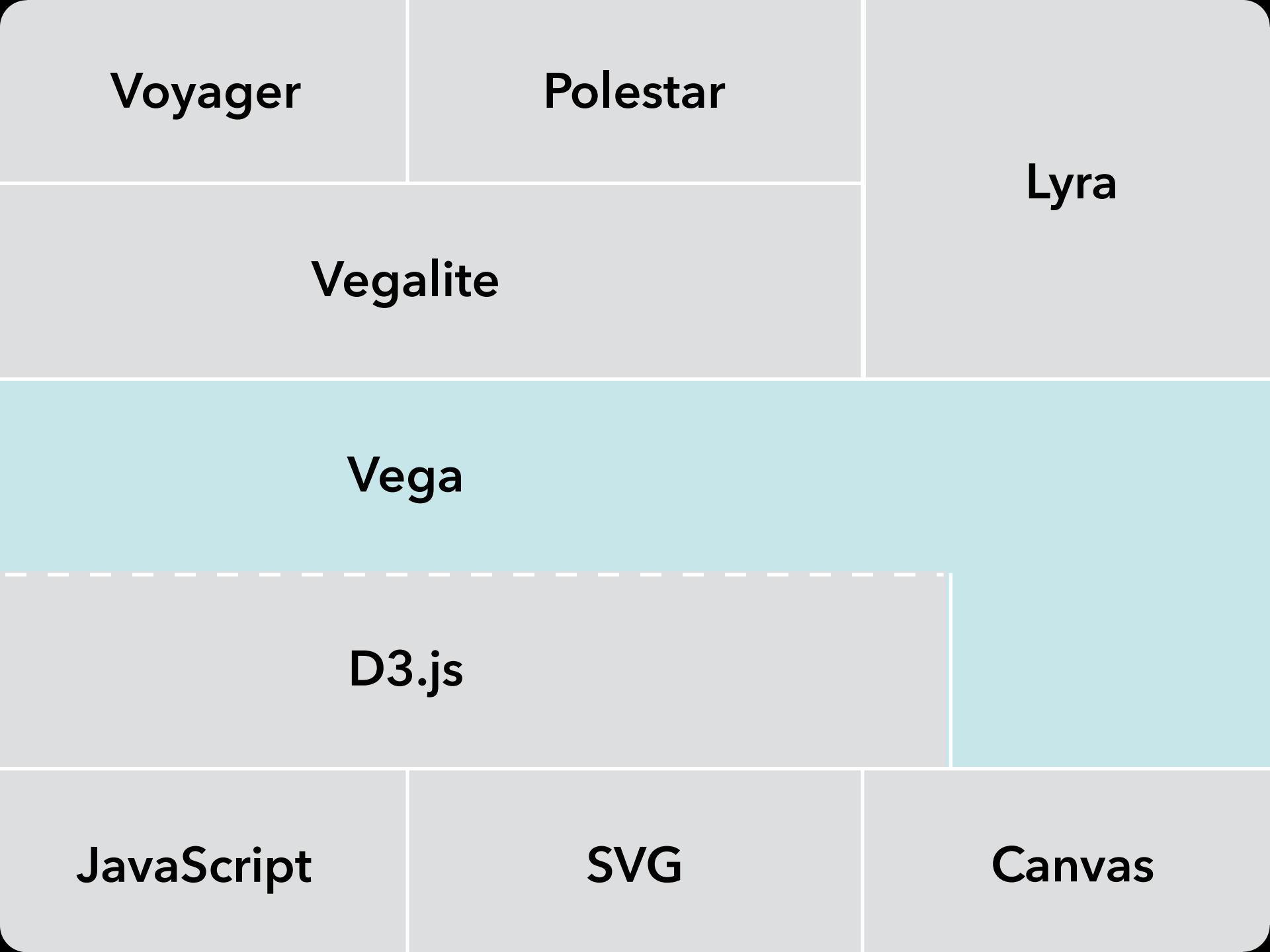

Voyager
Polestar
Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas

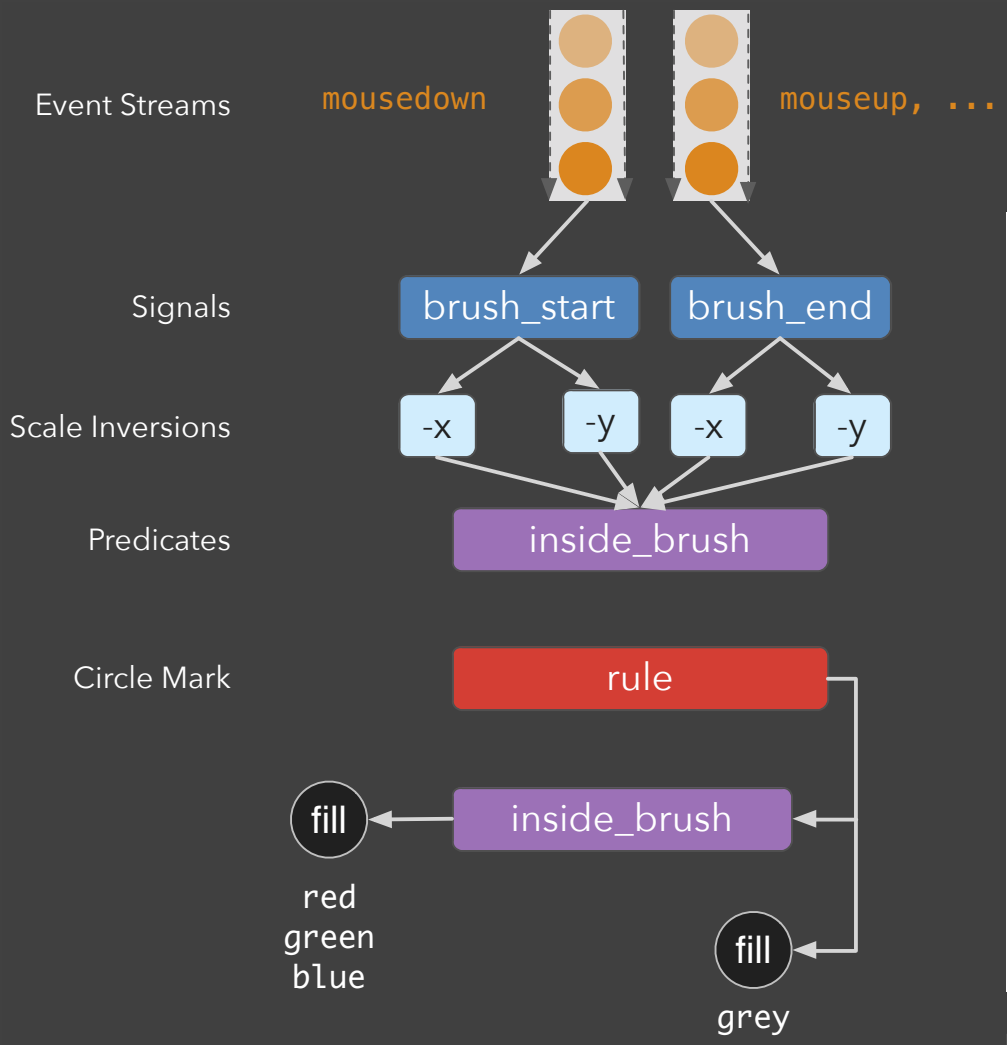

Voyager
Polestar
Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas

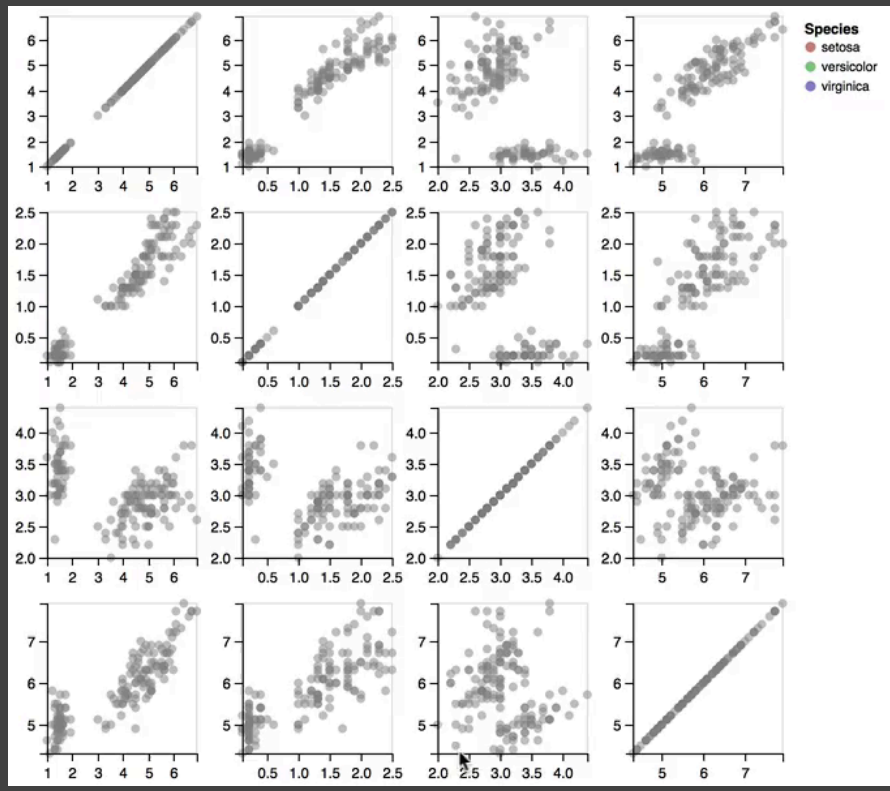
One last thing…

What about
interaction?


Voyager
Polestar
Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas

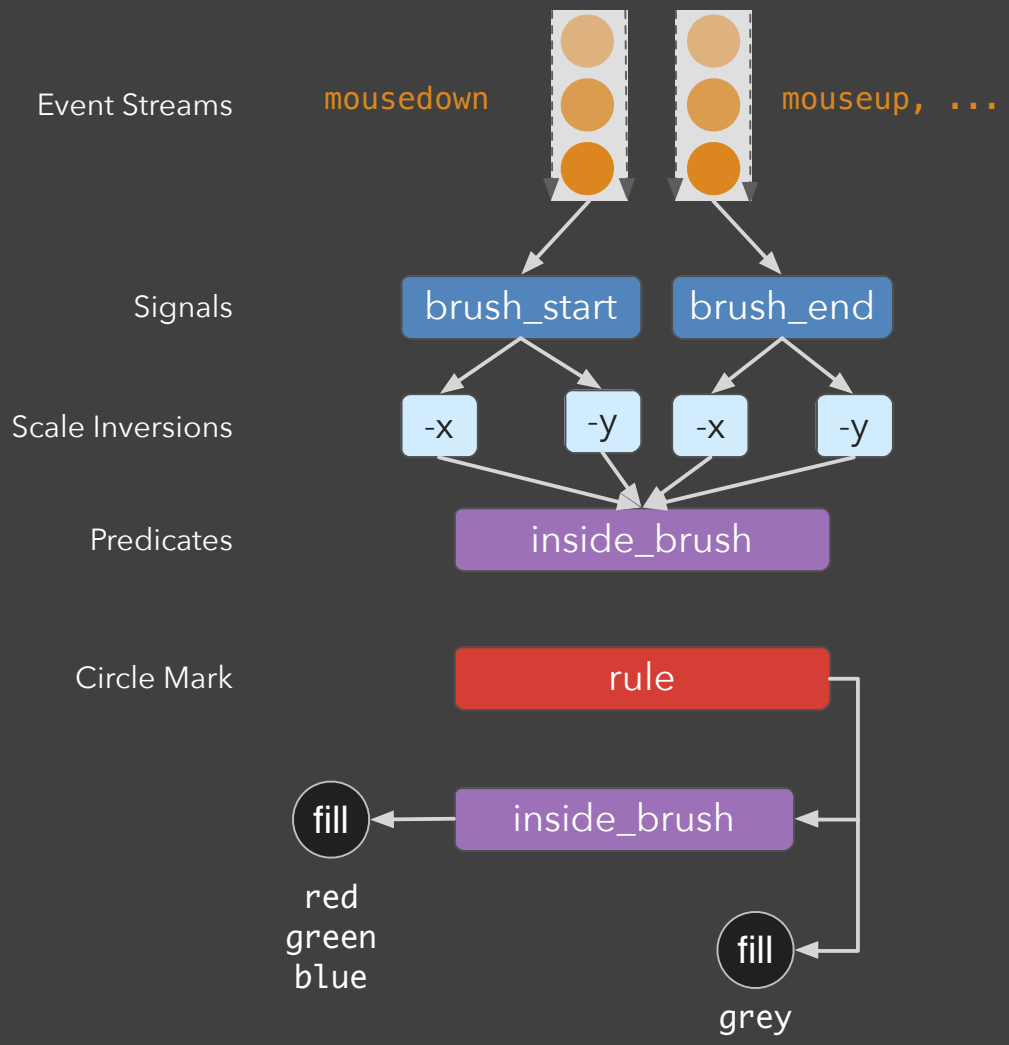

Voyager
Polestar
Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas

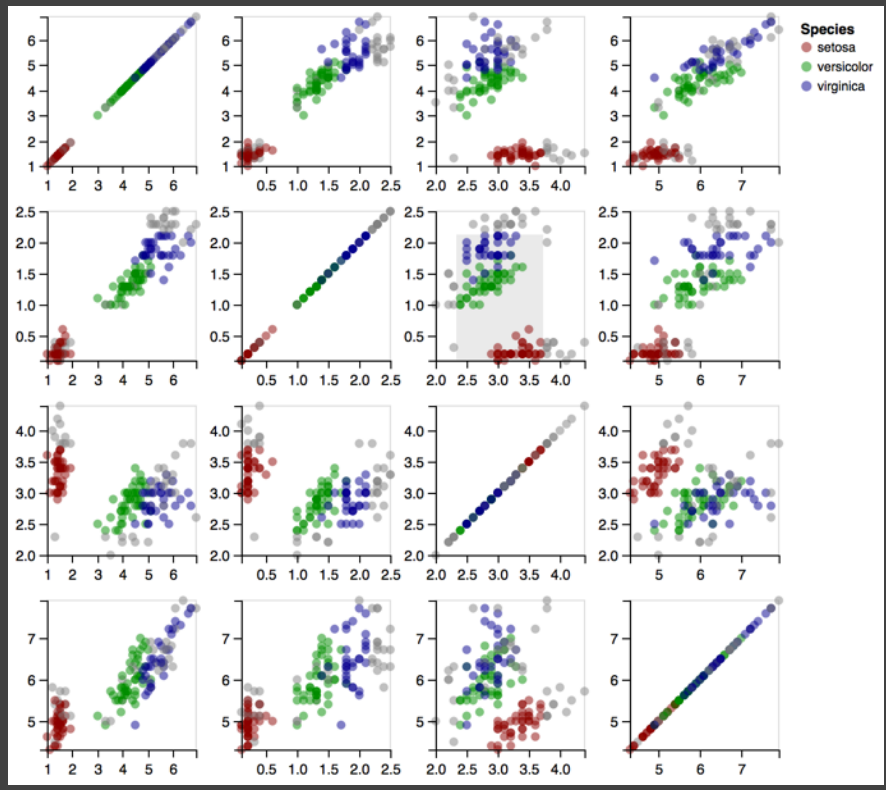




Reactive Vega

Satyanarayan et al. [UIST'14]





Key Insight: **Treat user input as first-class streaming data**

Adapt methods from functional reactive programming

Vega 2.0 (“Reactive Vega”)

Single declarative model for specifying
visual encodings + **interaction techniques**

Vega 2.0 (“Reactive Vega”)

Single declarative model for specifying **visual encodings** + **interaction techniques**

JSON –> Reactive Dataflow Graph

Designed ground-up for **streaming data**

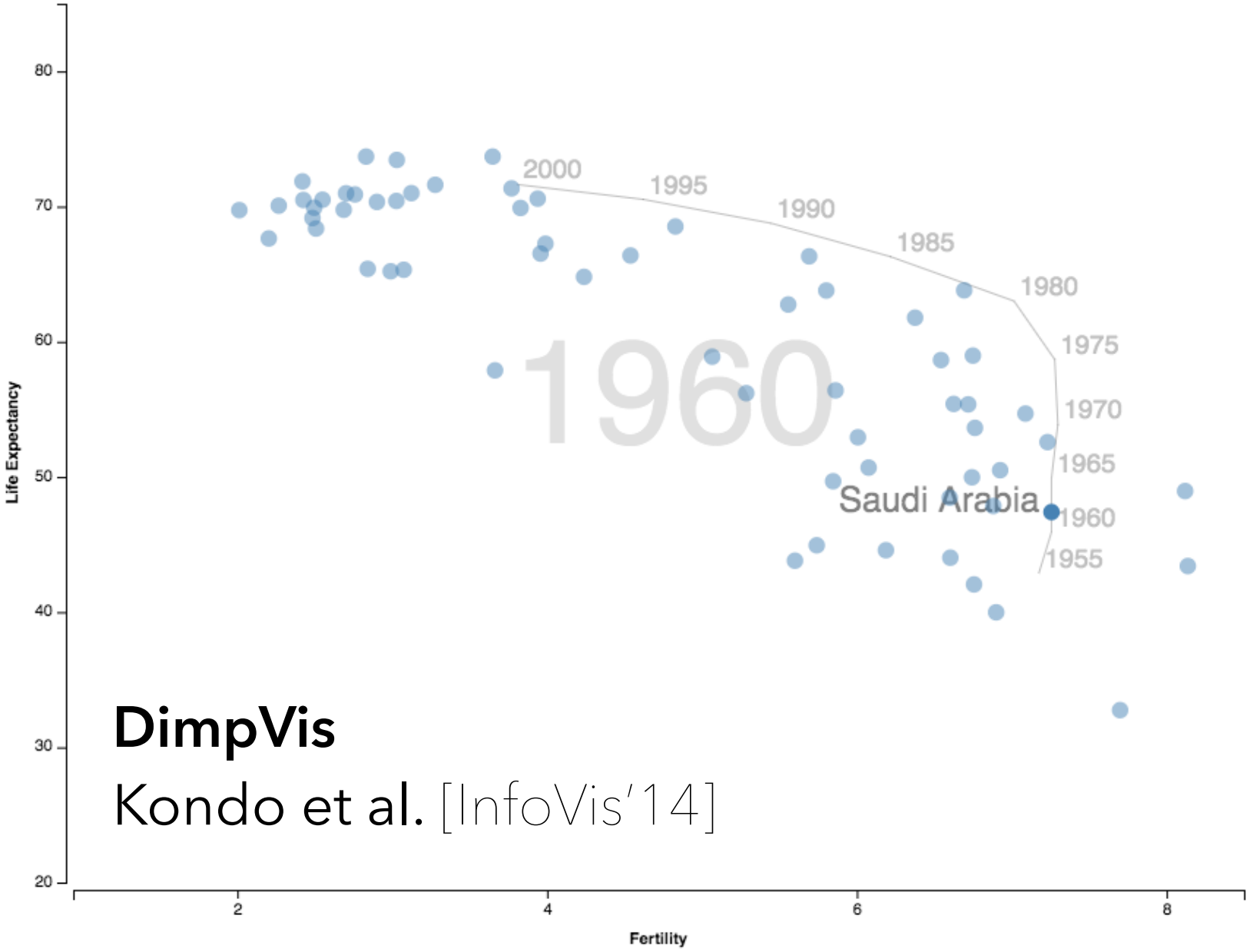
Performance matches or exceeds D3

Vega 2.0 (“Reactive Vega”)

Single declarative model for specifying
visual encodings + **interaction techniques**

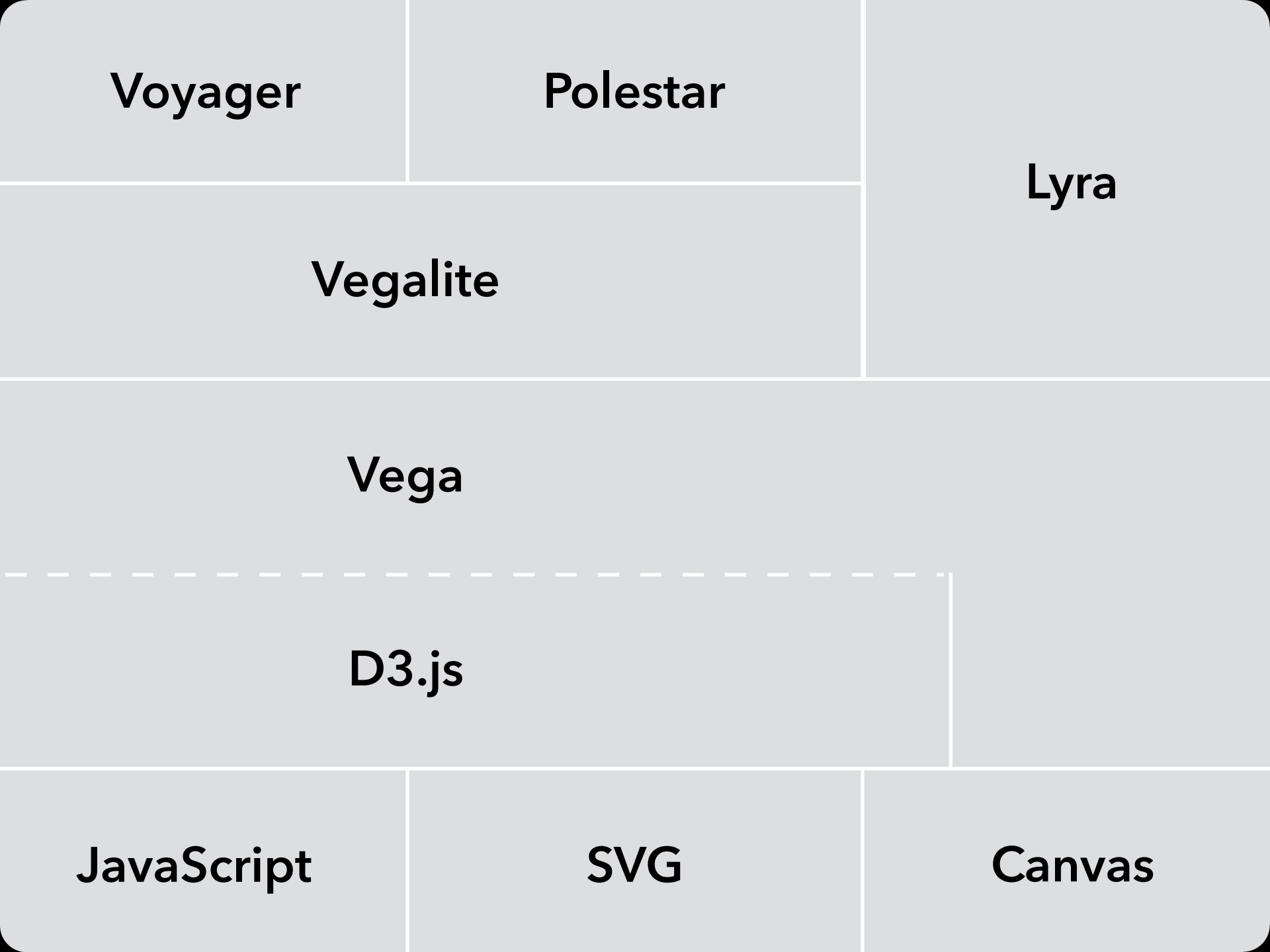
JSON –> Reactive Dataflow Graph
Designed ground-up for **streaming data**
Performance matches or exceeds D3

Portable
Client (*browser*) or server-side (*node.js*)
Pick your renderer: *Canvas*, *SVG*, …
Pick your input: *mouse*, *touch*, …



DimpVis

Kondo et al. [InfoVis'14]


Voyager
Polestar
Lyra
Vegalite
Vega
D3.js
JavaScript
SVG
Canvas

Open Challenges

Designing interactions interactively
How to convey + depict interactions?

Enhancing the “gallery” experience
Rapid assessment of multiple graphics
Embedding large views in small spaces?

Improving visualization recommenders
Learning from users, domain adaptation

Open Challenges

Designing interactions interactively
How to convey + depict interactions?

Enhancing the “gallery” experience
Rapid assessment of multiple graphics
Embedding large views in small spaces?

Improving visualization recommenders
Learning from users, domain adaptation

Debugging, debugging, debugging…

All Open Source…

Vega

https://vega.github.io

Vega A VISUALIZATION GRAMMAR

Vega is a declarative format for creating, saving, and sharing visualization designs. With Vega, visualizations are described in JSON, and generate interactive views using either HTML5 Canvas or SVG.

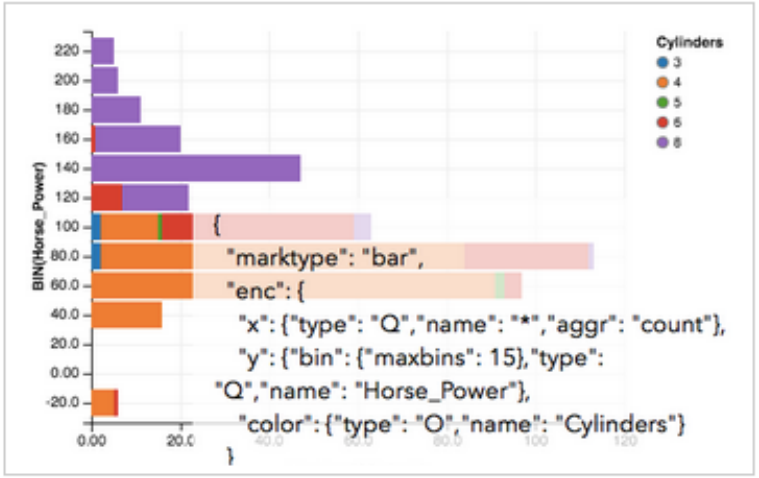
TOOLKITS

VEGA offers a full declarative visualization grammar, suitable for expressive custom interactive visualization design and programmatic generation.

Tutorial | Documentation | Discussion Forum

v1.5 (stable): download, examples, github

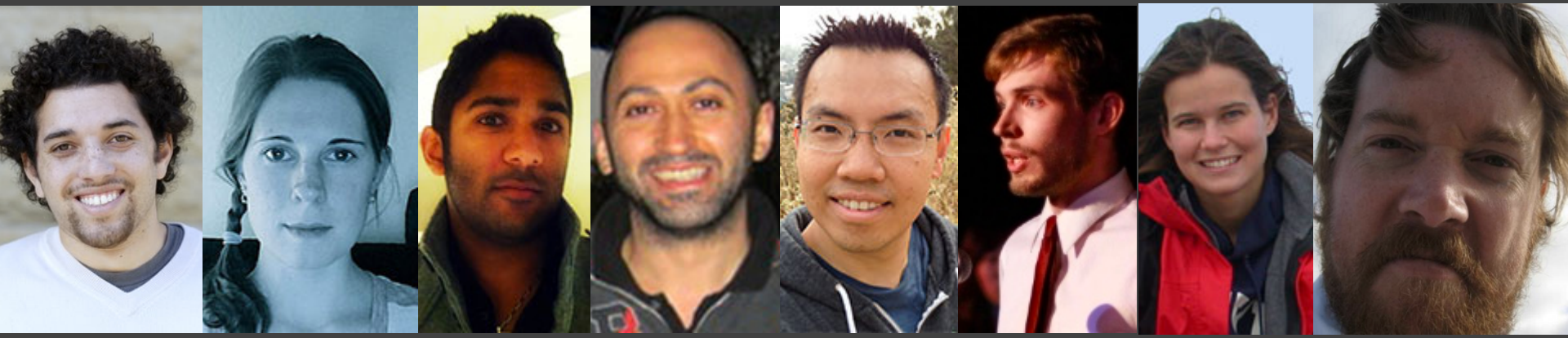
NEW v2.0 (dev): download, examples, github



NEW **VEGALITE** provides a higher-level grammar for visual analysis, comparable to ggplot or Tableau, that generates complete Vega specifications.

Online Editor | GitHub

vega.github.io



ISTC BIG DATA

DARPA

GORDON AND BETTY MOORE FOUNDATION

ISTC BIG DATA

DARPA

GORDON AND BETTY MOORE FOUNDATION

Raising the Bar (Chart)

THE NEXT GENERATION OF VISUALIZATION TOOLS

Jeffrey Heer @jeffrey_heer

http://vega.github.io/