Raising the Bar (Chart)
THE NEXT GENERATION OF VISUALIZATION TOOLS

Jeffrey Heer  @jeffrey_heer
Univ. of Washington + Trifacta
Visualizing Big Data!
Stratified Sampling
Binned Aggregation
imMens: Real-Time Visual Querying of Big Data
with Z. Liu & B. Jiang
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!
NICOLAS GARCIA BELMONTE
WEBGL FOR GRAPHICS AND DATA VISUALIZATION

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.
LANE HARRISON
USER-CENTERED VISUALIZATION RESEARCH
12:00 - 12:30
3:00 - 3:30

NIGEL HOLMES
USING HUMOR TO INFORM
Visualization Tools
Raising the Bar (Chart)

THE NEXT GENERATION OF VISUALIZATION TOOLS

Jeffrey Heer  @jeffrey_heer
Univ. of Washington + Trifacta
How might we create visualizations?
<table>
<thead>
<tr>
<th>task</th>
<th>row</th>
<th>col</th>
<th>d</th>
<th>d1</th>
<th>d2</th>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.73568</td>
<td>0.73568</td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>2</td>
<td>0.63512</td>
<td>0.73568</td>
<td>0.73568</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>3</td>
<td>0.59295</td>
<td>0.73119</td>
<td>0.73119</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>4</td>
<td>0.45864</td>
<td>0.24612</td>
<td>0.24612</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>5</td>
<td>0.49399</td>
<td>0</td>
<td>0.8119</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>6</td>
<td>0.85906</td>
<td>0.73568</td>
<td>0.8119</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>7</td>
<td>0.81534</td>
<td>0.73119</td>
<td>0.8119</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>8</td>
<td>0.71854</td>
<td>0.24612</td>
<td>0.8119</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>9</td>
<td>0.57947</td>
<td>0</td>
<td>0.8417</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>10</td>
<td>0.91546</td>
<td>0.73568</td>
<td>0.8417</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>11</td>
<td>0.91988</td>
<td>0.73119</td>
<td>0.8417</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>12</td>
<td>0.77741</td>
<td>0.24612</td>
<td>0.8417</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>13</td>
<td>0.29483</td>
<td>0</td>
<td>0.2825</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>14</td>
<td>0.75357</td>
<td>0.73568</td>
<td>0.2825</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>15</td>
<td>0.73206</td>
<td>0.73119</td>
<td>0.2825</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>SA</td>
<td>1</td>
<td>16</td>
<td>0.53588</td>
<td>0.24612</td>
<td>0.2825</td>
<td></td>
</tr>
</tbody>
</table>
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
```javascript
var svg = d3.select("svg")
  .attr("width", w)
  .attr("height", h)
  .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

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

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
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.

```javascript
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); })
```
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
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

Visual Analysis Grammars
VizQL, ggplot2, *Vegalite*

Visualization Grammars
Protovis, D3.js, *Vega*

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
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

Visual Analysis Grammars
VizQL, ggplot2, *Vegalite*

Visualization Grammars
Protovis, D3.js, *Vega*

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2, **Vega**

Visualization Grammars
Protovis, D3.js, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Charting Tools

Declarative Languages

Programming Toolkits
Visual Analysis Grammars
VizQL, ggplot2, **Vega**

Visualization Grammars
Protovis, D3.js, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Declarative Languages

Programming Toolkits
Interactive Data Exploration
Tableau, Lyra, Polestar, Voyager

Visual Analysis Grammars
VizQL, ggplot2, Vegalite

Visualization Grammars
Protovis, D3.js, Vega

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
D3.js

JavaScript

SVG

Canvas

Vega

VegaLite

Voyager

Polestar

Lyra
Visualization Grammar
Visualization Grammar

Data

Input data to visualize
## Visualization Grammar

<table>
<thead>
<tr>
<th>Data</th>
<th>Input data to visualize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transforms</td>
<td>Grouping, stats, projection, layout</td>
</tr>
</tbody>
</table>
## Visualization Grammar

<table>
<thead>
<tr>
<th><strong>Data</strong></th>
<th>Input data to visualize</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transforms</strong></td>
<td>Grouping, stats, projection, layout</td>
</tr>
<tr>
<td><strong>Scales</strong></td>
<td>Map data values to visual values</td>
</tr>
</tbody>
</table>
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

[Images of different marks: Area, Rect, Symbol, Image, Line, Text, Rule, Arc]
MARKS: Graphical Primitives
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.
The Lyra Visualization Design Environment (VDE) alpha
Arvind Satyanarayan, Kanit “Ham” Wongsuphasawat, Jeffrey Heer

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

based on the Railway Timetable by E. J. Marey
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"},
    "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"}
  }
}
D3.js

Vega

Vegalite

JavaScript

SVG

Canvas

Lyra
Polestar

A graphical interface for Vegalite

Rapid visualization via drag-and-drop

Named in honor of Polaris, the research project that led to Tableau.
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
1. Select **data variables**
2. Apply **transformations**
3. Pick visual **encodings**

**User**

**Voyager**
Visualiza­tion Browser

**Compass**
Recommenda­tion Engine

Data Schema & Statistics
Constrain & rank choices by data type, statistics & perceptual principles.

Compass
Recommendation Engine

Data Schema & Statistics

Voyager
Visualization Browser

User
Voyager Visualization Browser

Compass
Recommendation Engine

Data Schema & Statistics

Ranked and Clustered Vegalite Specifications

Voyager
Visualization Browser

User

Vegalite Compiler

Vegalite Specifications
Voyager Visualization Browser

Compass Recommendation Engine

Ranked and Clustered Vegalite Specifications

Data Schema & Statistics

Voyager Visualization Browser

Interactive Visualizations

Vega Renderer

Vega Specifications

Vega Specifications

Vegalite Compiler
Voyager Visualization Browser

Compass Recommendation Engine

Data Schema & Statistics

Ranked and Clustered Vegalite Specifications

User

Interactive Visualizations

Voyager Visualization Browser

Vega Renderer

Vega Specifications

Vegalite Compiler

Vegalite Specifications

Interactive Visualizations
Voyager Visualization Browser

Interactive Visualizations

User Selection

Compass Recommendation Engine

Ranked and Clustered Vegalite Specifications

Vega Renderer

User Selection, Data Schema & Statistics

Voyager Visualization Browser

Interactive Visualizations

Vega Specifications

Vegalite Compiler

Vegalite Specifications
Implements data coverage!
+3x variable sets shown
+1.5x more interacted with.
One last thing...
What about interaction?
Event Streams
- mousedown
- mouseup, ...

Signals
- brush_start
- brush_end
- -x
- -y

Scale Inversions
- -x
- -y

Predicates
- inside_brush

Circle Mark
- rule

- fill
- inside_brush
- fill
- grey

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]
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 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.

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

Tutorial | Documentation | Discussion Forum

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

NEW v2.0 (dev): download, examples, github
Raising the Bar *(Chart)*

THE NEXT GENERATION OF VISUALIZATION TOOLS

Jeffrey Heer  @jeffrey_heer

http://vega.github.io/