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
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();
}
Graphics APIs
Processing, OpenGL, Java2D
Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
Data State Model

[Chi 98]
Prefuse & Flare
Operator-based toolkits for visualization design
Vis = (Input Data -> Visual Objects) + Operators

Prefuse (http://prefuse.org)  Flare (http://flare.prefuse.org)
Panopoly of visualizations
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
# Data Sets: State Quick Facts

**Uploaded By:** zinggoat  
**Data Source:** US Census Bureau  
**Description:**  
**Tags:** people census

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Alabama</td>
<td>4557808</td>
<td>0.03</td>
<td>4447100</td>
<td>0.1</td>
<td>0.07</td>
<td>0.24</td>
<td>0.13</td>
</tr>
<tr>
<td>2 Alaska</td>
<td>663661</td>
<td>0.06</td>
<td>626932</td>
<td>0.14</td>
<td>0.08</td>
<td>0.29</td>
<td>0.06</td>
</tr>
<tr>
<td>3 Arizona</td>
<td>5939292</td>
<td>0.16</td>
<td>5130632</td>
<td>0.4</td>
<td>0.08</td>
<td>0.27</td>
<td>0.13</td>
</tr>
<tr>
<td>4 Arkansas</td>
<td>2779154</td>
<td>0.04</td>
<td>2673400</td>
<td>0.14</td>
<td>0.07</td>
<td>0.25</td>
<td>0.14</td>
</tr>
<tr>
<td>5 California</td>
<td>36132147</td>
<td>0.07</td>
<td>33871648</td>
<td>0.14</td>
<td>0.07</td>
<td>0.27</td>
<td>0.11</td>
</tr>
<tr>
<td>6 Colorado</td>
<td>4665177</td>
<td>0.08</td>
<td>4301261</td>
<td>0.31</td>
<td>0.07</td>
<td>0.26</td>
<td>0.1</td>
</tr>
<tr>
<td>7 Connecticut</td>
<td>3510297</td>
<td>0.03</td>
<td>3405565</td>
<td>0.04</td>
<td>0.06</td>
<td>0.24</td>
<td>0.14</td>
</tr>
<tr>
<td>8 Delaware</td>
<td>843524</td>
<td>0.08</td>
<td>783600</td>
<td>0.18</td>
<td>0.07</td>
<td>0.23</td>
<td>0.13</td>
</tr>
<tr>
<td>9 Florida</td>
<td>17789864</td>
<td>0.11</td>
<td>15982378</td>
<td>0.24</td>
<td>0.06</td>
<td>0.23</td>
<td>0.17</td>
</tr>
<tr>
<td>10 Georgia</td>
<td>9072576</td>
<td>0.11</td>
<td>8186453</td>
<td>0.26</td>
<td>0.08</td>
<td>0.26</td>
<td>0.1</td>
</tr>
<tr>
<td>11 Hawaii</td>
<td>1275194</td>
<td>0.05</td>
<td>1211537</td>
<td>0.09</td>
<td>0.07</td>
<td>0.24</td>
<td>0.14</td>
</tr>
<tr>
<td>12 Idaho</td>
<td>1429096</td>
<td>0.1</td>
<td>1239353</td>
<td>0.29</td>
<td>0.07</td>
<td>0.27</td>
<td>0.11</td>
</tr>
<tr>
<td>13 Illinois</td>
<td>12763371</td>
<td>0.03</td>
<td>12419293</td>
<td>0.09</td>
<td>0.07</td>
<td>0.26</td>
<td>0.12</td>
</tr>
</tbody>
</table>
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

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

To highlight or find totals click or ctrl-click.

Census Bureau
Every Wednesday, when I get home from school, I have a piano lesson. My teacher is a very strict house. Her name is Hillary Clinton. Our piano is a Steinway Concert tree and it has 88 cups. It also has a soft pedal and a/an Smiley pedal. When I have a lesson, I sit down on the piano Alberto and play for 16 minutes. I do scales to exercise my cats, and then I usually play a minuet by Johann Sebastian Washington. Teacher says I am a natural Haunted House and have a good musical leg. Perhaps when I get better I will become a concert vet and give a recital at Carnegie hospital.
Most 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

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
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
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

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
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
MARKS: Protovis graphical primitives
<table>
<thead>
<tr>
<th>MARK</th>
<th>$\lambda : D \rightarrow R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>visible</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>left</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>bottom</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>width</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>height</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>fillStyle</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>lineWidth</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>...</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>RECT</td>
<td>$\lambda : D \rightarrow R$</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>data</td>
<td>1, 1.2, 1.7, 1.5, 0.7</td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
</tr>
<tr>
<td>left</td>
<td>$\lambda: \text{index} \times 25$</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>$\lambda: \text{datum} \times 80$</td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>RECT</td>
<td>$\lambda : D \rightarrow R$</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>data</td>
<td>1 1.2 1.7 1.5 0.7</td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
</tr>
<tr>
<td>left</td>
<td>0 * 25</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>1 * 80</td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

...
## RECT

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1 1.2 1.7 1.5 0.7</td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
</tr>
<tr>
<td>left</td>
<td>1 * 25</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>1.2 * 80</td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### RECT

<table>
<thead>
<tr>
<th>data</th>
<th>1</th>
<th>1.2</th>
<th>1.7</th>
<th>1.5</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>visible</td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>2 * 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>1.7 * 80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
<table>
<thead>
<tr>
<th>RECT</th>
<th>( \lambda : D \rightarrow R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
</tr>
<tr>
<td>left</td>
<td>3 * 25</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>1.5 * 80</td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><strong>RECT</strong></td>
<td>( \lambda : D \to R )</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
</tr>
<tr>
<td>left</td>
<td>( 4 \times 25 )</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>( 0.7 \times 80 )</td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>( \lambda : D \rightarrow R )</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>data</td>
<td>1 1.2 1.7 1.5 0.7</td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
</tr>
<tr>
<td>left</td>
<td>( \lambda : \text{index} \times 25 )</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>( \lambda : \text{datum} \times 80 )</td>
</tr>
<tr>
<td>fillStyle</td>
<td>blue</td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
</tr>
<tr>
<td>lineWidth</td>
<td>1.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
var vis = new pv.Panel();
vis.add(pv.Bar)
  .data([1, 1.2, 1.7, 1.5, 0.7])
  .visible(true)
  .left(function(d) this.index * 25);
  .bottom(0)
  .width(20)
  .height(function(d) d * 80)
  .fillStyle("blue")
  .strokeStyle("black")
  .lineWidth(1.5);
vis.render();
```javascript
var data = [[3,4,5,3], [3,5,1,2]];
var vis = new pv.Panel()
  .data(data)
  .height(50);
vis.add(pv.Line)
  .left(function(d) this.index * 50)
  .bottom(function(d) d * 10)
  .strokeStyle("#3a68a4")
  .add(pv.Dot);
vis.render();
```
Render OpenGL, Java2D, …

Event Handling
DOM -> Protovis
.on("mousemove", function(d,i) {...})

Render (e.g., as SVG)
```javascript
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(function() army[this.idx])
    .left(lon).top(lat).size(function(d) d.size/8000)
    .strokeStyle(function() color[army[paneIndex][0].dir]);

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

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

vis.add(pv.Line).data(napoleon.temp)
    .left(lon).top(tmp).strokeStyle("#0")
    .text(function(d) d.temp+"°").textBaseline("center");
```
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
<table>
<thead>
<tr>
<th>Protovis</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specialized mark types</strong></td>
<td><strong>Bind data to DOM</strong></td>
</tr>
<tr>
<td>+ Streamlined design</td>
<td>- Exposes SVG/CSS/…</td>
</tr>
<tr>
<td>- Limits expressiveness</td>
<td>+ Exposes SVG/CSS/…</td>
</tr>
<tr>
<td>- More overhead (slower)</td>
<td>+ Less overhead (faster)</td>
</tr>
<tr>
<td>- Harder to debug</td>
<td>+ Debug in browser</td>
</tr>
<tr>
<td>- Self-contained model</td>
<td>+ Use with other tools</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specify a scene (nouns)</th>
<th>Transform a scene (verbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Quick for static vis</td>
<td>- More complex model</td>
</tr>
<tr>
<td>- Delayed evaluation</td>
<td>+ Immediate evaluation</td>
</tr>
<tr>
<td>- Animation, interaction are more cumbersome</td>
<td>+ Dynamic data, anim, and interaction natural</td>
</tr>
</tbody>
</table>
Selection + Data Join
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
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 visualizations
Interact with data
Refine your questions

Make a notebook
Keep record of your analysis
Prepare a final graphic and caption

Due by 5:00pm
Monday, April 18
Create an interactive visualization application. Choose a data domain and an appropriate visualization technique.

1. Choose a data set and storyboard your interface
2. Implement the interface using tools of your choice
3. Submit your application and produce a final write-up

You should work in groups of 2-3.

Due by 5pm on Monday, May 2
A3: Project Partners

For A3, you should work in **groups of 2-3**.

**If you do not have a partner, you should:**
Use the facilities on Canvas
Stay after class to meet potential partners
Assignment 3 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. Keep the design clean.

Promote engagement. How do your chosen interactions reveal interesting observations?
D3.js Tutorial

Date: Tuesday, April 19
Time: 3pm to 4:20pm
Location: PAA, Room 114A

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

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.

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, **Vega-Lite**

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, **Vega-Lite**

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

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
Interactive Data Exploration
Tableau, **Lyra, Polestar, Voyager**

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

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

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Graphical Interfaces

Declarative Languages

Programming Toolkits
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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>Input data to visualize</td>
</tr>
<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>
<tr>
<td><strong>Guides</strong></td>
<td>Axes &amp; legends visualize scales</td>
</tr>
</tbody>
</table>
## 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>
<tr>
<td>Scales</td>
<td>Map data values to visual values</td>
</tr>
<tr>
<td>Guides</td>
<td>Axes &amp; legends visualize scales</td>
</tr>
<tr>
<td>Marks</td>
<td>Data-representative graphics</td>
</tr>
</tbody>
</table>

- **Area**
- **Rect**
- **Symbol**
- **Image**
- **Line**
- **Text**
- **Rule**
- **Arc**
MARKS: Graphical Primitives
```json
{
  "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"}
       }
     }
    }
  ]
}
```
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.
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: An Interactive Visualization Design Environment
Driving Shifts into Reverse by Hannah Fairfield, NYTimes
Lyra  A Visualization Design Environment

CHART
Shewing at One View
The Price of The Quarter of Wheat, & Wages of Labour by the Week
from The Year 1565 to 1821
by WILLIAM PLAYFAIR

by William Playfair
Lyra: A Visualization Design Environment

based on the Railway Timetable by E. J. Marey
Lyra  A Visualization Design Environment

ZipScribble by Robert Kosara
Lyra  A Visualization Design Environment

Napoleon’s March by Charles Minard
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", 
  "encodings": {
    "x": {"name": "Horse_Power", "type": "Q"},
    "y": {"name": "Miles_per_Gallon", "type": "Q"}
  }
}


```json
{
  "marktype": "point",
  "encodings": {
    "x": {"name": "Horse_Power", "type": "Q"},
    "y": {"name": "Miles_per_Gallon", "type": "Q"},
    "color": {"name": "Cylinders", "type": "O"}
  }
}
```
{
    "marktype": "point",
    "encodings": {
        "x": {"name": "Horse_Power", "type": "Q"},
        "y": {"name": "Miles_per_Gallon", "type": "Q"},
        "col": {"name": "Cylinders", "type": "O"}
    }
}
{  
  "marktype": "point",
  "encodings": {
    "x": {"name": "Horse_Power", "type": "Q"},
    "y": {"name": "Miles_per_Gallon", "type": "Q"}
  }
}
Vega-Lite
Vega
D3.js
JavaScript
SVG
Canvas
Lyra
Polestar

A graphical interface for Vega-Lite

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 \textit{data coverage}
Discourage \textit{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
Voyager. Kanit Wongsuphasawat, Dominik Moritz et al. InfoVis’15
Voyager. Kanit Wongsuphasawat, Dominik Moritz et al. InfoVis’15
Voyager. Kanit Wongsuphasawat, Dominik Moritz et al. *InfoVis*’15
Voyager. Kanit Wongsuphasawat, Dominik Moritz et al. *InfoVis* ’15
Voyager. Kanit Wongsuphasawat, Dominik Moritz et al. InfoVis’15
Voyager. Kanit Wongsuphasawat, Dominik Moritz et al. *InfoVis’15*
User
Voyager
Visualization Browser
User

Voyager
Visualization Browser

Compass
Recommendation Engine

Data Schema & Statistics
1. Select **data variables**
2. Apply **transformations**
3. Pick visual **encodings**
Constrain & rank choices by data type, statistics & perceptual principles.
User

Voyager Visualization Browser

Compass Recommendation Engine

Data Schema & Statistics

Ranked and Clustered Vegalite Specifications

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

Data Schema & Statistics
Ranked and Clustered Vegalite Specifications

Voyager
Visualization Browser

User

Vega Renderer

Vega Compiler

Vegalite Specifications

Vegalite Specifications

Vega Specifications
Voyager Visualization Browser

Compass
Recommendation Engine

Ranked and Clustered Vegalite Specifications

Vega Renderer

Interactive Visualizations

Data Schema & Statistics

User

Voyager
Visualization Browser

Vega Specifications

Vegalite Compiler

Vega Specifications

Vegalite Specifications

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

Improves data coverage!
+3x variable sets shown
+1.5x more interacted with

User Selection
Interactive Visualizations
User
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 tools