Visualization Tools

Matthew Conlen  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

$$\text{Vis} = (\text{Input Data} \rightarrow \text{Visual Objects}) + \text{Operators}$$

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
Chart Typologies
## Data Sets: State Quick Facts

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

### People QuickFacts

<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>1293953</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

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

Data files:
- Retail sales per capita 2002
- Minority-owned firms percent of total 1997
- Women-owned firms percent of total 1997
- Housing units authorized by building permits 2004

Full image:
- Federal spending 2004 ($1000)
- Land area 2000 (square miles)
- Persons per square mile 2000
- FIPS Code

Comments: This data set has not yet been rated.
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
The Grammar of Graphics, 1999
Chart Typologies
Excel, Many Eyes, Google Charts

Component Architectures
Prefuse, Flare, Improvise, VTK

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

Visual Analysis Grammars
VizQL, ggplot2

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
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

Ease-of-Use

Expressiveness
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, Jeff Heer
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

**Data**  
Input data to visualize

**Transforms**  
Grouping, stats, projection, layout

**Scales**  
Map data values to visual values
## Visualization Grammar

<table>
<thead>
<tr>
<th></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

Data
Input data to visualize

Transforms
Grouping, stats, projection, layout

Scales
Map data values to visual values

Guides
Axes & legends visualize scales

Marks
Data-representative graphics
MARKS: Protovis graphical primitives
<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></td>
<td></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>$\lambda: \text{index} \times 25$</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>data</td>
<td>1</td>
</tr>
<tr>
<td>---------</td>
<td>-----</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>
</tbody>
</table>

...
<table>
<thead>
<tr>
<th>RECT</th>
<th>( \lambda : D \rightarrow R )</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>
<tr>
<td>RECT</td>
<td>$\lambda : D \rightarrow R$</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>$2 \times 25$</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>$1.7 \times 80$</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>
</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>3 * 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>1.5 * 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>

$\lambda: D \rightarrow R$
<table>
<thead>
<tr>
<th>RECT</th>
<th>( \lambda : D \rightarrow R )</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>4 * 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 * 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>

Data rectangle with fill style blue and stroke style black.
<table>
<thead>
<tr>
<th>RECT</th>
<th>$\lambda : D \rightarrow R$</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>$\lambda$: index * 25</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>$\lambda$: datum * 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>
```javascript
var vis = new pv.Panel();
vis.add(pv.Bar)
  .data([1, 1.2, 1.7, 1.5, 0.7])
  .visible(true)
  .left((d) => this.index * 25);
  .bottom(0)
  .width(20)
  .height((d) => d * 80)
  .fillStyle("blue")
  .strokeStyle("black")
  .lineWidth(1.5);
vis.render();
```
var army = pv.nest(napoleon.army, "dir", "group");
var vis = new pv.Panel();

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

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

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

vis.add(pv.Line).data(napoleon.temp)
  .left(lon).top(tmp).strokeStyle("#0")
  .add(pv.Label)
  .top((d) => 5 + tmp(d))
  .text((d) => d.temp+"° " +d.date.substr(0,6))
  .textAlign("center").textBaseline("top").font("italic 10px Georgia");
Obesity Map | Vadim Ogievetsky
d3.js Data-Driven Documents

with Mike Bostock, Jason Davies, Vadim Ogievetsky, Jeff Here
Protovis

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

Specify a scene (nouns)
  + Quick for static vis
  - Delayed evaluation
  - Animation, interaction are more cumbersome
Protovis

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

*Specify a scene (nouns)*
  + Quick for static vis
  - Delayed evaluation
  - Animation, interaction are more cumbersome

D3

*Bind data to DOM*
  - Exposes SVG/CSS/…
  + Exposes SVG/CSS/…
  + Less overhead (faster)
  + Debug in browser
  + Use with other tools

*Transform a scene (verbs)*
  - More complex model
  + Immediate evaluation
  + Dynamic data, anim, and interaction natural
D3 Selections

The core abstraction in D3 is a selection.
D3 Selections

The core abstraction in D3 is a selection.

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

// add new SVG to page body
D3 Selections

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

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

// Select & update existing rectangles contained in the SVG element
svg.selectAll("rect")
    .attr("width", 100)        // set rect widths to 100px
    .style("fill", "steelblue"); // set rect fill colors
```
Data Binding

Selections can *bind* data and DOM elements.

```javascript
var values = [ {…}, {…}, {…}, … ];  // input data as JS objects
```
Data Binding

Selections can bind data and DOM elements.

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

// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars").data(values);
```
Data Binding

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

```javascript
var values = [ {…}, {…}, {…}, … ]; // input data as JS objects
// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars").data(values);

// What if the DOM elements don’t exist yet? The enter set represents data
// values that do not yet have matching DOM elements.
bars.enter().append("rect").attr("class", "bars");
```
Selections can **bind** data and DOM elements.

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

// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars").data(values);

// What if the DOM elements don’t exist yet? The enter set represents data values that do not yet have matching DOM elements.
bars.enter().append("rect").attr("class", "bars");

// What if data values are removed? The exit set is a selection of existing DOM elements who no longer have matching data values.
bars.exit().remove();
```
The Data Join

DATA VALUES

ENTER
Data values without matching DOM elements.

UPDATE
Existing DOM elements, bound to valid data.

EXIT
DOM elements whose bound data has gone “stale”.

ELEMENTS
The Data Join

\[ \text{var } s = \text{d3.selectAll(...).data(...)} \]

**ENTER**

Data values without matching DOM elements.
\[ s.\text{enter().append(...)} \]

**UPDATE**

Existing DOM elements, bound to valid data.
\[ s \]

**EXIT**

DOM elements whose bound data has gone "stale".
\[ s.\text{exit()} \]
D3 Modules

Data Parsing / Formatting (JSON, CSV, …)
Shape Helpers (arcs, curves, areas, symbols, …)
Scale Transforms (linear, log, ordinal, …)
Color Spaces (RGB, HSL, LAB, …)
Animated Transitions (tweening, easing, …)
Geographic Mapping (projections, clipping, …)
Layout Algorithms (stack, pie, force, trees, …)
Interactive Behaviors (brush, zoom, drag, …)

Many of these correspond to future lecture topics!
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

Deliverable: A sequence of annotated visualizations that clearly communicate your findings.

Due by 11:59pm Tonight!
Tutorials

Introduction to **Web Development & D3.js**

*Next Lecture*
A3: Interactive Prototype

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

Due by 11:59pm on Monday, February 10.

Work in project teams of 3-5 people.
Requirements

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

**Web-based.** D3 is encouraged, but not required. Deploy your visualization using GitHub pages.

**Write-up.** Provide design rationale on your web page.
A3 & Final Project Team

Form a **team of 3-5** for A3 and the Final Project. Start thinking about your Final Project, too! A3 is open-ended, but you can use it to start exploring your FP topic if you like.

Submit signup form by **Friday 1/31, 11:59pm**.

**If you do not have team mates**, you should:
- Use the facilities on Canvas/Piazza
- [piazza.com/washington/winter2020/cse442](https://piazza.com/washington/winter2020/cse442)
- Stay after class to meet potential partners
Team Member Roles

We encourage you to structure team responsibilities!

**Coordinator**: Organize meetings, track deadlines, etc.

**Data Lead**: Data wrangling, management, distillation

**Tech Lead**: Manage code integration, GitHub repo

**UX Lead**: Visualization/interaction design & evaluation

One may have multiple roles, share work across roles…
Interactive Prototype Tips

**Start now.** It will take longer than you think.

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

**Promote engagement.** How do your chosen interactions reveal interesting observations?
Interactive Prototype Tips

Prototyping is a valuable activity. Feel free to use this exercise to your advantage for the final project.

**Final Project**
Interactive dashboard for non-expert users.

(Politics, Sports, Climate, Finance, Entertainment, …)

**NEW!** Explorable explanation of technical topic.

(Algorithm, Theorem, Science, …)
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?

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.
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
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

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

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, Vega-Lite

Visualization Grammars
Protovis, D3.js, Vega

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D
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
Voyager. Wongsuphasawat et al. InfoVis’15, CHI’17
User
User → Data Set → Voyager
   Visualization Browser
User

Voyager
Visualization Browser

CompassQL
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.
Voyager Visualization Browser

CompassQL Recommendation Engine

Data Schema & Statistics

Ranked and Clustered Vega-Lite Specifications

Voyager Visualization Browser

Interactive Visualizations

Vega-Lite Specifications

Vega Renderer

User Data Schema & Statistics

Vega-Lite Compiler

Vega Specifications
Voyager Visualization Browser

- Interactive Visualizations
- User-Defined Focus View

CompassQL
Recommendation Engine

- Focus View, Data Schema & Statistics
- Ranked and Clustered Vega-Lite Specifications

Voyager
Visualization Browser

- Interactive Visualizations
- Vega-Lite Specifications

Vega
Renderer

- Vega Specifications

Vega-Lite
Compiler

User

User-Defined Focus View

Interactive Visualizations
**Implements data coverage!**

+ $4x$ variable sets shown
+ $2x$ more interacted with

**CompassQL**
Recommendation Engine

Focus View, Data Schema & Statistics

Ranked and Clustered Vega-Lite Specifications

**Voyager**
Visualization Browser

Interactive Visualizations

Vega-Lite Specifications

**Vega**
Renderer

**Vega-Lite**
Compiler

User-Defined Focus View

Interactive Visualizations

User
Evaluating Interactive Articles

Embedding data in two dimensions

The same brightness feature can be used to position the artworks in 2D space instead of 1D. The pieces have more room to spread out.

On the right you see a simple 2-dimensional embedding based on image brightness, but this isn’t the only way to position the artworks. In fact, there are many, and some projections are more useful than others.

Use the slider to vary the influence that the brightness and artwork age have in determining the embedding positions. As you move the slider from brightness to artwork age, the embedding changes from highlighting bright and dark images, and starts to cluster recent modern-day images in the bottom left corner whereas older artworks are moved farther away (hover over images to see their date).

Artwork Age

Brightness

The embedding you see here is actually a linear 1D embedding, whose resulting scatter is then mapped on a space-filling Hilbert curve to give the illusion of a 2D
Dimensionality Reduction

http://www.ggobi.org/
Principal Components Analysis

1. Mean-center the data.
2. Find $\perp$ basis vectors that maximize the data variance.
3. Plot the data using the top vectors.
PCA of Genomes [Demiralp et al. ‘13]
Many Reduction Techniques!

**General Strategies:**
Matrix Factorization  
Nearest Neighbor (Topological) Methods

**Popular Techniques:**
Principal Components Analysis (PCA)  
t-Dist. Stochastic Neighbor Embedding (t-SNE)  
Uniform Manifold Approx. & Projection (UMAP)
The Beginner's Guide to Dimensionality Reduction

Explore the methods that data scientists use to visualize high-dimensional data.

By: Matthew Conlen and Fred Hohman
July 16, 2018

Dimensionality reduction is a powerful technique used by data scientists to look for hidden structure in data. The method is useful in a number of domains, for example document categorization, protein disorder prediction, and machine learning model debugging.[2]

The results of a dimensionality reduction algorithm can be visualized to reveal patterns and clusters of similar or dissimilar data. Even though the data is displayed in only two or three dimensions, structures present in higher dimensions are maintained, at least roughly.[7]

The technique is available in many applications, for
How to Use t-SNE Effectively

Although extremely useful for visualizing high-dimensional data, t-SNE plots can sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.
Visualizing t-SNE [Wattenberg et al. '16]
Time Curves [Bach et al. ‘16]

Timeline:
1 2 3 4 5 6 7
Time difference
Circles are data cases with a time stamp. Similar colors indicate similar data cases.

Folding:
1
2 3 4 5 6 7

Time curve:
1 2 3 4 5 6 7
Similarity
The temporal ordering of data cases is preserved. Spatial proximity now indicates similarity.

(a) Folding time

Wikipedia “Chocolate” Article

U.S. Precipitation over 1 Year