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)
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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>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>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>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>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>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>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>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>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>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>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>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>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>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 word 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
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
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
Visualization Grammar
Visualization Grammar

Data

Input data to visualize
Visualization Grammar

Data
Input data to visualize

Transforms
Grouping, stats, projection, layout
# Visualization Grammar

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Input data to visualize</td>
</tr>
<tr>
<td>Transforms</td>
<td>Grouping, stats, projection, layout</td>
</tr>
<tr>
<td>Scales</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
MARKS: Protovis graphical primitives
<table>
<thead>
<tr>
<th>MARK</th>
<th>$\lambda : D \to R$</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>strokeLine</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>lineWidth</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>...</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>Attribute</td>
<td>Value</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>
</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>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>
<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>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 * 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 * 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>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>strokeLineStyle</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>$\lambda : D \rightarrow R$</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>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>

...
### 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>( \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((d) => this.index * 25);
  .bottom(0)
  .width(20)
  .height((d) => d * 80)
  .fillStyle("blue")
  .strokeStyle("black")
  .lineWidth(1.5);
vis.render();
```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(() => army[this.idx])
  .left(lon).top(lat).size((d) => d.size/8000)
  .strokeStyle((d) => 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)
  .lineWidth(1).strokeStyle("#ccc")
  .anchor("right").add(pv.Label)
  .font("italic 10px Georgia")
  .text((d) => d+"°").textBaseline("center");

vis.add(pv.Line).data(napoleon.temp)
  .left(lon).top(tmp).strokeStyle("#0")
  .add(pv.Label)
  .top((d) => 5 + tmp(d))
  .text((d) => d.temp+"° "+d.date.substr(0,6))
  .textAlign("center").textBaseline("top").font("italic 10px Georgia");
```
PRELUDE NO.1 IN C MAJOR, BWV 846
(FROM WELL-TEMPERED CLAVIER, BOOK 1)

BY J.S. BACH

Bach’s Prelude #1 in C Major | Jieun Oh
d3.js  Data-Driven Documents

with Mike Bostock & Vadim Ogievetsky
Protovis

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

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

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

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

D3

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

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

The core abstraction in D3 is a selection.
D3 Selections

The core abstraction in D3 is a *selection*.

```javascript
// Add and configure an SVG element
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
// set SVG width to 500px
// set SVG height to 300px
D3 Selections

The core abstraction in D3 is a `selection`.

```javascript
var svg = d3.append("svg")
    .attr("width", 500)
    .attr("height", 300);

svg.selectAll("rect")
    .attr("width", 100)
    .style("fill", "steelblue");
```

// Add and configure an SVG element

// Add new SVG to page body
// set SVG width to 500px
// set SVG height to 300px

// Select & update existing rectangles contained in the SVG element

// select all SVG rectangles
// set rect widths to 100px
// 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");
```
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");

// 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

ENTER
Data values without matching DOM elements.

UPDATE
Existing DOM elements, bound to valid data.

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

DATA VALUES

ELEMENTS
The Data Join

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

**ENTER**
Data values without matching DOM elements.
`s.enter().append(...)`

**UPDATE**
Existing DOM elements, bound to valid data.
`s`

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

**Create visualizations:**
- Interact with data
- Refine your questions

**Author a report:**
- Screenshots of most insightful views (10+)
- Include titles and captions for each view

Due by 11:59pm Friday, Apr 13

DONE
D3.js Tutorial

Date: Thursday, April 19
Time: 4:30pm to 6:30pm
Location: Sieg 134

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

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

Due by 11:59pm on Monday, April 30.

Work in project teams of 3-4 people.
Requirements

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

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

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

Form a **team of 3-4** for A3 ASAP.

(Start thinking about your Final Project, too!)

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

Submit signup form by **Friday 4/20, 11:59pm**.

**If you do not have team mates**, you should:
- Use the facilities on Canvas
- Stay after class/tutorial to meet potential partners
Interactive Prototype Tips

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

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

**Promote engagement.** How do your chosen interactions reveal interesting observations?
A Visualization Tool Stack
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

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

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

Charting Tools

Declarative Languages

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); })
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?

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
Interactive Data Exploration
Tableau, Lyra, 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
VEGA-LITE
A Grammar of Interactive Graphics

Kanit "Ham" Wongsuphasawat @kanitw
Dominik Moritz @domoritz
Arvind Satyanarayan @arvindsatya1
Jeffrey Heer @jeffrey_heer

Interactive Data Lab @uwdata
University of Washington
Grammar of Graphics for Customized Designs

Offer fine-grained control for composing interactive graphics.

But require verbose specifications and technical expertise.
Grammar of Graphics for Exploration

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

Infer sensible defaults and allow customization by overriding defaults.

But limited support for interactions.
How might we author *interactive* graphics in the midst of analysis?
Vega-Lite: A Grammar of Interactive Graphics
A. Satyanarayan, D. Moritz, K. Wongsuphasawat & J. Heer. TVCG 2017
Vega-Lite: Scatter Plot

{  
  "data": {"url": "data/cars.json"},  
  "mark": "point",  
  "encoding": {  
    "x": {"field": "Horse_Power", "type": "Q"},  
    "y": {"field": "Miles_per_Gallon", "type": "Q"}  
  }  
}
Vega-Lite: Scatter Plot

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

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

```
{
  "data": {"url": "data/cars.json"},
  "mark": "point",
  "encoding": {
    "x": {"field": "Horse_Power", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"}
  }
}
```
Vega-Lite: 2D Histogram

```
{
  "data": {"url": "data/cars.json"},
  "mark": "point",
  "encoding": {
    "x": {"field": "Horse_Power", "type": "Q", "bin": true},
    "y": {"field": "Miles_per_Gallon", "type": "Q", "bin": true},
    "size": {"field": "\*", "type": "Q", "aggregate": "count"}
  }
}
```
RESEARCH GOAL:
Extend grammars of statistical graphics to enable multi-view composition and interaction.
Vega-Lite: A Grammar of Graphics
Vega-Lite: A Grammar of Multi-View Graphics
Vega-Lite: A Grammar of Interactive Multi-View Graphics
```json
{
  "data": {
    "url": "data/flights.json",
    "mark": "bar",
    "encoding": {
      "x": {
        "field": "hour",
        "bin": true,
        "type": "Q"
      },
      "y": {
        "field": "*",
        "aggregate": "count",
        "type": "Q"
      }
    }
  }
}
```


```json
{
    "repeat": {"column": ["hour", "delay", "distance"]},
    "spec": {
        "data": {"url": "data/flights.json"},
        "mark": "bar",
        "encoding": {
            "x": {"field": {"repeat": "column"}, "bin": true, "type": "Q"},
            "y": {"field": "*", "aggregate": "count", "type": "Q"}
        }
    }
}
```
```json
{
    "repeat": "column": ["hour", "delay", "distance"],
    "spec": {
        "data": {"url": "data/flights.json"},
        "mark": "bar",
        "encoding": {
            "x": {"field": {"repeat": "column"}, "bin": true, "type": "Q"},
            "y": {"field": ":", "aggregate": "count", "type": "Q"}
        }
    }
}
```
```json
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [
      {
        "data": {"url": "data/flights.json"},
        "mark": "bar",
        "encoding": {
          "x": {"field": {"repeat": "column"}, "bin": true, "type": "Q"},
          "y": {"field": "+", "aggregate": "count", "type": "Q"}
        }
      },
      ...
    ],
    "encoding": {
      ...
    },
    "color": {"value": "goldenrod"}
  }
}
```
can be initialized
```json
{
  "repeat": {
    "column": ["hour", "delay", "distance"]
  },
  "spec": {
    "layers": [{
      "select": {
        "region": {
          "type": "interval",
          "project": {
            "channels": ["x"]
          }
        }
      }
    }]
  }
}
```
{  "repeat": {"column": ["hour", "delay", "distance"]},  "spec": {  "layers": [{  "select": {  "region": {  "type": "interval", "project": {"channels": ["x"]}, ...  }  },  },  {  "transform": {"filterWith": "region"}  }]}}
Interactive Selections
Altair is a declarative statistical visualization library for Python, based on Vega-Lite.

With Altair, you can spend more time understanding your data and its meaning. Altair’s API is simple, friendly and consistent and built on top of the powerful Vega-Lite visualization grammar. This elegant simplicity produces beautiful and effective visualizations with a minimal amount of code.
How might we support more effective data exploration?
Common analysis pitfalls:
Overlook data quality issues
Fixate on specific relationships
Plus many other cognitive biases

[Heuer 1999, Kahneman 2011, …]
Voyager: Combine Manual Specification with Visualization Recommenders
**Key Idea:** Augment manual exploration with visualization recommendations sensitive to the user’s current focus.

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

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

Candidate charts are pruned and ranked using models of estimated perceptual effectiveness.
Compared to existing tools, leads to **over 4x more variable sets seen**, and **over 2x more variable sets interacted with**.

“**The related view suggestion accelerates exploration a lot.**”

“I like that it shows me what fields to include in order to see a specific graph. Otherwise, I have to do a lot of trial and error and can't express what I wanted to see.”

“These related views are so good but it’s also spoiling that I start thinking less. I’m not sure if that’s really a good thing.”
Voyager: Combine Manual Specification with Visualization Recommenders

WORK IN PROGRESS...

Voyager integration with JupyterLab!