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
OpenGL, Java2D, GDI+, Processing
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();
}
US Air Traffic, Aaron Koblin
Graphics APIs
OpenGL, Java2D, GDI+, Processing
Component Model Architectures
Prefuse, Flare, Improvise

Graphics APIs
OpenGL, Java2D, GDI+, Processing
Data State Model
[Chi 98]
Prefuse & Flare

Operator-based toolkits for visualization design

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

Prefuse (http://prefuse.org)  Flare (http://flare.prefuse.org)
Component Model Architectures
Prefuse, Flare, Improvise

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

Component Model Architectures
Prefuse, Flare, Improvise

Graphics APIs
OpenGL, Java2D, GDI+, Processing
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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<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>2</td>
<td>Alaska</td>
<td>663861</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</td>
<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>4</td>
<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>5</td>
<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>6</td>
<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>7</td>
<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>8</td>
<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>9</td>
<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>10</td>
<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>11</td>
<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>12</td>
<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>13</td>
<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 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
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 Smily 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.

Chart Typologies
Excel, Many Eyes, Google Charts

Component Model Architectures
Prefuse, Flare, Improvise

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

Visual Analysis Languages
Tableau VizQL, ggplot2

Component Model Architectures
Prefuse, Flare, Improvise

Graphics APIs
OpenGL, Java2D, GDI+, Processing
Polaris
Research at Stanford by Stolte, Tang, and Hanrahan.
Polaris / Tableau Approach

Insight: can simultaneously specify both database queries and visualization

Choose data, then visualization, not vice versa

Use smart defaults for visual encodings

More recently: automate visualization design
Specifying Table Configurations

Operands are the database fields
- Each operand interpreted as a set {...}
- Quantitative and Ordinal fields treated differently

Three operators:
- concatenation (+)
- cross product (x)
- nest (/)
Table Algebra: Operands

**Ordinal fields**: interpret domain as a set that partitions table into rows and columns.

Quarter = \{((Qtr1),(Qtr2),(Qtr3),(Qtr4))\}

<table>
<thead>
<tr>
<th>Qtr1</th>
<th>Qtr2</th>
<th>Qtr3</th>
<th>Qtr4</th>
</tr>
</thead>
<tbody>
<tr>
<td>95892</td>
<td>101760</td>
<td>105282</td>
<td>98225</td>
</tr>
</tbody>
</table>

**Quantitative fields**: treat domain as single element set and encode spatially as axes:

Profit = \{((Profit[-410,650]))\}
Concatenation (+) Operator

Ordered union of set interpretations

Quarter + Product Type
= \{(Qtr1),(Qtr2),(Qtr3),(Qtr4)\} + \{(Coffee), (Espresso)\}
= \{(Qtr1),(Qtr2),(Qtr3),(Qtr4),(Coffee),(Espresso)\}

Profit + Sales = \{(Profit[-310,620]),(Sales[0,1000])\}
Cross (x) Operator

Cross-product of set interpretations

Quarter x Product Type

= \{ (Qtr1, Coffee), (Qtr1, Tea), (Qtr2, Coffee), (Qtr2, Tea), (Qtr3, Coffee), (Qtr3, Tea), (Qtr4, Coffee), (Qtr4, Tea) \}

Product Type x Profit =
Nest (/) Operator

Cross-product filtered by existing records

Quarter x Month
creates twelve entries for each quarter. i.e., (Qtr1, December)

Quarter / Month
creates three entries per quarter based on tuples in database (not semantics)
Table Algebra

The operators (+, x, /) and operands (O, Q) provide an *algebra* for tabular visualization.

Algebraic statements are then mapped to:

- **Queries** - selection, projection, group-by aggregation
- **Visualizations** - trellis plot partitions, visual encodings

In Tableau, users make statements via drag-and-drop

Note that this specifies operands *NOT* operators!

Operators are inferred by data type (O, Q)
Querying the Database

1. Select records from the database, filtering by user-defined criteria.
2. Partition the records into layers and panes. The same record may appear in multiple partitions.
3. Group, sort, and aggregate the relations within each pane.
4. Render and compose layers.
### Ordinal - Ordinal

<table>
<thead>
<tr>
<th>State</th>
<th>Coffee</th>
<th>Espresso</th>
<th>Herbal Tea</th>
<th>Tea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colorado</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Connecticut</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Florida</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Illinois</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Iowa</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Louisiana</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Massachusetts</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td></td>
</tr>
<tr>
<td>Missouri</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Nevada</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>New Hampshire</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>New Mexico</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Ohio</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Oklahoma</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Oregon</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Texas</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Utah</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Washington</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
</tbody>
</table>
Quantitative - Quantitative
Ordinal - Quantitative
Grammar of Graphics [Wilkinson, Wickham]

Data  Input data to visualize
Operators  Grouping, statistics, layout
Coordinates  Cartesian & polar coordinates
Scales  Map data values to visual values
Guides  Axes & legends visualize scales
Marks  Geometric primitives
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 Languages
Tableau VizQL, ggplot2

Component Model Architectures
Prefuse, Flare, Improvise

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

Visual Analysis Languages
Tableau VizQL, ggplot2

Component Model Architectures
Prefuse, Flare, Improvise

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

Visual Analysis Languages
Tableau VizQL, ggplot2

Component Model Architectures
Prefuse, Flare, Improvise

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

Visual Analysis Languages
Tableau VizQL, ggplot2

Component Model Architectures
Prefuse, Flare, Improvise

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

Visual Analysis Languages
Tableau VizQL, ggplot2

Visualization Grammars
Protovis, D3.js, Vega

Component Model Architectures
Prefuse, Flare, Improvise

Graphics APIs
OpenGL, Java2D, GDI+, Processing
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.

Protovis: A Language 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>BAR</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>
BAR $\lambda: D \rightarrow R$

<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></td>
<td>true</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>0 * 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 * 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>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\begin{itemize}
\item[] \textbf{BAR} \hfill $\lambda : D \rightarrow R$
\item[] \begin{tabular}{|l|c|c|c|c|c|}
\hline
\text{data} & 1 & 1.2 & 1.7 & 1.5 & 0.7 \\
\hline
\text{visible} & true & \\
\hline
\text{left} & 1 \times 25 & \\
\hline
\text{bottom} & 0 & \\
\hline
\text{width} & 20 & \\
\hline
\text{height} & 1.2 \times 80 & \\
\hline
\text{fillStyle} & blue & \\
\hline
\text{strokeStyle} & black & \\
\hline
\text{lineWidth} & 1.5 & \\
\hline
\text{...} & ... & \\
\hline
\end{tabular}
\end{itemize}

\begin{itemize}
\item[] \textbf{BART}
\item[] \textbf{BAR}$\lambda$ \textbf{BAR}$\lambda$
\end{itemize}
<table>
<thead>
<tr>
<th></th>
<th></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>data</td>
<td>1</td>
<td>1.2</td>
<td>1.7</td>
<td>1.5</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>visible</td>
<td>true</td>
<td></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>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
<td></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>
<td></td>
</tr>
<tr>
<td>height</td>
<td>1.7 * 80</td>
<td></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>
<td></td>
</tr>
<tr>
<td>strokeStyle</td>
<td>black</td>
<td></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>
<td></td>
</tr>
</tbody>
</table>

\[ \lambda : D \rightarrow R \]
<table>
<thead>
<tr>
<th>BAR</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>$3 \times 25$</td>
</tr>
<tr>
<td>bottom</td>
<td>0</td>
</tr>
<tr>
<td>width</td>
<td>20</td>
</tr>
<tr>
<td>height</td>
<td>$1.5 \times 80$</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>BAR</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>$4 \times 25$</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>
</tbody>
</table>

The diagram shows a bar chart with the following specifications: 1 column, 5 bars, data range from 0.7 to 2.5.
<table>
<thead>
<tr>
<th><strong>BAR</strong></th>
<th>( \lambda: D \to 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>( \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();
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();
Event Handling
DOM -> Protovis
.on("mousemove",
function(d,i) {...})
Productivity - *Faster Design Cycle, Less Code*
Comparison: 5x less code, 10x less dev time

Portability - *Multiple Implementations*
JavaScript, Adobe Flash, Java/JVM

Performance - *Optimization* (in Protovis-Java)
Just-in-time compilation; parallel execution
Hardware accelerated rendering
Up to **20x scalability boost** over prior toolkits
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.Label).data(napoleon.temp)
  .left(lon).top(tmp).strokeStyle("#0")
  .add(pv.Label)
  .top(function(d) 5 + tmp(d))
  .text(function(d) d.temp+"° "+d.date.substr(0,6))
  .textBaseline("top").font("italic 10px Georgia");
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/…
- Less overhead (faster)
- Debug in browser
- Use with other tools

**Transform a scene (verbs)**
- More complex model
- Immediate evaluation
- Dynamic data, animation, and interaction natural
Selection + Data Join
Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Languages
Tableau VizQL, ggplot2

Visualization Grammars
Protovis, D3.js, Vega

Component Model Architectures
Prefuse, Flare, Improvise

Graphics APIs
OpenGL, Java2D, GDI+, Processing
Administrivia
Assignment 2: 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 wiki notebook

• Keep record of your analysis
• Prepare a final graphic and caption

Due by 5:00pm
Monday, Jan 27
A3: Interactive Visualization

Create an interactive visualization application. Choose a data domain and select 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.

Due by 5pm on Monday, February 10
Assignment 3: Project Partners

For A3, you should work in groups of 2.

If you do not have a partner, you should
1) Use the facilities on Piazza
2) Stay after class to meet potential partners
Assignment 3 Tips

1) Start now. It will take longer than you think.

2) Keep it simple. (Eschew the kitchen sink.) Choose the minimal set of interactions that enables users to explore and generate interesting insights. Keep the design clean.

3) Promote engagement. How do your chosen interactions reveal interesting observations?
Comparing Approaches
No Humble Pie
Pie Chart: Processing

```java
size(150, 150); background(255);
smooth(); noStroke();
int diameter = 140;
float[] data = {1, 1.2, 1.7, 1.5, .7};
float lastAng = 0, sum = 0;
for (int i = 0; i < data.length; i++)
    sum += data[i];
for (int i = 0; i < data.length; i++) {
    fill(data[i] * 120);
    float ang = data[i] / sum * 2 * PI;
    arc(width / 2, height / 2, diameter, diameter,
        lastAng, lastAng + ang);
    lastAng += ang;
}
```
Pie Chart: Flare

var data:Data = Data.fromArray(
    [{value:1}, {value:1.2}, {value:1.7}, ...]
);
var vis:Visualization = new Visualization(data);

vis.bounds = new Rectangle(0, 0, 150, 150);
vis.operators.add(new PieLayout("data.value", 0.9));
vis.operators.add(new ColorEncoder("data.value", "nodes", "fillColor");

vis.update();
addChild(vis);
Pie Chart: Protovis

var vis = new pv.Panel()
  .width(150)
  .height(150)
  .add(pv.Wedge)
  .left(75)
  .bottom(75)
  .outerRadius(70)
  .angle(function(d) { return d * 2 * Math.PI; })
  .data(pv.normalize([1, 1.2, 1.7, 1.5, .7]));

vis.render();
Pie Chart: D3

```javascript
var data = [1, 1.2, 1.7, 1.5, .7];
var color = d3.scale.category20();
var svg = d3.select("body").append("svg")
  .attr("width", 150)
  .attr("height", 150);
var g = svg.append("g")
  .attr("transform", "translate(75, 75)"");
var arcs = g.selectAll("path")
  .data(d3.layout.pie().sort(null)(data))
  .enter().append("path")
  .style("fill", function(d,i) { return color(i); })
  .attr("d", d3.svg.arc().innerRadius(0).outerRadius(70));
```
Vega & Lyra
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, and then explore Vega visualizations using the web-based Vega Editor.
MARKS: Vega graphical primitives
Vega 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
Vega Design Goals

Balancing rapid + expressive design
Protovis-style marks, data transforms & layouts

Reuse & sharing of visualization designs
JSON format -> chart components or stand-alone vis

Support programmatic generation
Provides an intermediate specification language

Performance & platform flexibility
Browser or server-side; Canvas or SVG
For the highest level graphics (elegant, custom, expensive), enter the crunched data or the graphical output into Adobe Illustrator. ...This program gives complete control over typography, line weight, color, grids, layout--just what we need for doing graphical work.

It is a serious, complex design program; you may want to work with real graphical designers who will surely know their way around Illustrator.

Lyra: An Interactive Visualization Design Environment
The swing backward
The average number of miles that Americans drive annually begins to fall, so the curve appears to turn around.
### CHART

**Shewing at One View**

*The Price of The Quarter of Wheat, & Wages of Labour by the Week from The Year 1664 to 1811*

by WILLIAM PLAYFAIR

<table>
<thead>
<tr>
<th>Monarch</th>
<th>Start Year</th>
<th>End Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elizabeth</td>
<td>1565</td>
<td>1603</td>
</tr>
<tr>
<td>James I</td>
<td>1603</td>
<td>1625</td>
</tr>
<tr>
<td>Charles I</td>
<td>1625</td>
<td>1649</td>
</tr>
<tr>
<td>Cromwell</td>
<td>1649</td>
<td>1660</td>
</tr>
<tr>
<td>Charles II</td>
<td>1660</td>
<td>1680</td>
</tr>
<tr>
<td>William &amp; Mary</td>
<td>1680</td>
<td>1694</td>
</tr>
<tr>
<td>Anne</td>
<td>1694</td>
<td>1702</td>
</tr>
<tr>
<td>George I</td>
<td>1702</td>
<td>1714</td>
</tr>
<tr>
<td>George II</td>
<td>1714</td>
<td>1727</td>
</tr>
<tr>
<td>George III</td>
<td>1727</td>
<td>1820</td>
</tr>
<tr>
<td>George IV</td>
<td>1820</td>
<td>1830</td>
</tr>
</tbody>
</table>

**Data Pipelines**

- Pipeline name: Monarchs
- Import data from: monarchs2

**Extend Formulas**

- (name = "Cromwell") as commonwealth
- (name = "commonwealth" && end) as offset

**Visual Layers**

- Axes
  - Wheat
  - Year

- Marks
  - Monarchs
    - name: Monarchs
    - pipeline: Monarchs
  - Visual Transforms
    - x
      - start: [start]
      - width: 33
    - end
      - [end]
    - y
      - start: [start]
      - height: 8
      - end: 0
    - fill
      - color: Fill Color
      - commonwealth
    - opacity
    - stroke
      - color
      - width