Information Visualization

SWE 432, Fall 2018
Design and Implementation of Software for the Web
Today

• What is an information visualization?
• Lots of examples of visualizations
• Patterns and anti-patterns for visualization
• How can you build an information visualization d3.js?
Review: What is a software process?

• A structured set of activities required to develop a software product
  • Specification
  • Design and implementation
  • Validation
  • Evolution (operation and maintenance)
• Goal: Minimize Risk
  • Falling behind schedule
  • Changes to requirements
  • Bugs/unintended effects of changes
Review: Test-Stage-Production

Developer Environments

Testing Environment

Staging Environment

Beta/Dogfooding

User Requests

Production Environment

Revisions are “promoted” towards production
Review: Operations Responsibility

• Once we **deploy**, someone has to monitor software, make sure it’s running OK, no bugs, etc

• Assume 3 environments:
  • Test, Staging, Production

• Whose job is it?

<table>
<thead>
<tr>
<th>Waterfall</th>
<th>Developers</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile</td>
<td>Test</td>
<td>Staging</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production</td>
</tr>
<tr>
<td>DevOps</td>
<td>Test</td>
<td>Staging</td>
</tr>
<tr>
<td></td>
<td>Staging</td>
<td>Production</td>
</tr>
<tr>
<td></td>
<td>Production</td>
<td></td>
</tr>
</tbody>
</table>
Review: DevOps Values

- No silos, no walls, no responsibility "pipelines"
- One team owns changes "from cradle to grave"
- You are the support person for your changes, regardless of platform
- Example: Facebook mobile teams
Review: Deployment Example: Facebook.com

- Developers working in their own branch
  - ~1 week of development

- Weekly
  - All changes from week that are ready for release

- ~1 week of development

- Release Branch
  - 4 days
  - All changes that survived stabilizing

- Production
  - Twice Daily
    - Your change doesn’t go out unless you’re there that day at that time to support it!
    - “When in doubt back out”

- Stabilize
  - 3 days
  - All changes from week that are ready for release
Information visualization

• Technology has made data pervasive
  • health, finance, commerce, customer, travel, demographics, communications, ...
  • some of it “big”
• Information visualization: the use of interactive visual representations to amplify cognition
  • e.g., discover insights, answer questions
Cholera Epidemic in London, 1854

- >500 fatal attacks of cholera in 10 days
  - Concentrated in Broad Street area of London
  - Many died in a few hours
- Dominant theory of disease: caused by noxious odors
- Afflicted streets deserted by >75% inhabitants
Investigation and aftermath

• Based on visualization, did case by case investigation
• Found that 61 / 83 positive identified as using well water from Broad Street pump
• Board ordered pump-handle to be removed from well
• Epidemic soon ended
• Solved centuries old question of how cholera spread
Methods used by Snow

• Placed data in appropriate **context** for assessing cause & effect
  • Plotted on map, included well location
  • Reveals proximity as cause
• Made quantitative **comparisons**
  • Fewer deaths closer to brewery, could investigate cause
• Considered **alternative** explanations & contrary cases
  • Investigated cases not close to pump, often found connection to pump
• Assessment of possible **errors** in numbers
Charles Minard’s Map of Napoleon’s Russian Campaign of 1812
Chapel & Garofalo, Rock ‘N Roll is Here to Pay: The History and Politics of the Music Industry
What is an information visualization?

- Data -> Visual representation
  - Rows in data table -> elements in data visualization
    - e.g., historical person -> circle in visualization
  - Columns of data -> visual variables
    - e.g., relationship to another person -> edge in network visualization
Data transformations

- Classing / binning: Quantitative —> ordinal
  - Maps ranges onto **classes** of variables
  - Can also count # of items in each class w/ histogram
- Sorting: Nominal —> ordinal
  - Add order between items in sets
- Descriptive statistics: mean, average, median, max, min, …
Visual structures

• 3 components
  • spatial substrate
  • marks
  • marks’ graphical properties
Spatial substrate

• Axes that divide space
• Types of axes - unstructured, nominal, ordinal, quantitative
• Composition - use of multiple orthogonal axes (e.g., 2D scatterplot, 3D)
Folding

- continuing an axis by continuing in different space
Marks

- Points (0D)
- Lines (1D)
- Areas (2D)
- Volumes (3D)
Marks’ graphical properties

- Quantitative (Q), Ordinal (O), Nominal (N)
- Filled circle - good; open circle - bad

![Graphical properties diagram](image)
Effectiveness of graphical properties

- Quantitative (Q), Ordinal (O), Nominal (N)
- Filled circle - good; open circle - bad

<table>
<thead>
<tr>
<th>Spatial Extent</th>
<th>Spatial</th>
<th>Q</th>
<th>O</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Position)</td>
<td>Grayscale</td>
<td>Q</td>
<td>O</td>
<td>N</td>
</tr>
<tr>
<td>Size</td>
<td>Color</td>
<td>Q</td>
<td>O</td>
<td>N</td>
</tr>
<tr>
<td>Differential</td>
<td>Texture</td>
<td>Q</td>
<td>O</td>
<td>N</td>
</tr>
<tr>
<td>Orientation</td>
<td>Shape</td>
<td>Q</td>
<td>O</td>
<td>N</td>
</tr>
</tbody>
</table>
Examples of visualizations
Time-series data
Stacked graph

- Supports visual summation of multiple components
Small multiples

• supports separate comparison of data series
• may have better legibility than placing all in single plot
Maps
Choropleth map

- Groups data by area, maps to color
Cartograms

• Encodes two variables w/ size & color
Hierarchies
Node link diagram
Dendrogram

- leaf nodes of hierarchy on edges of circle
Treemaps
Some challenges in information visualizations

• Data binding
  • You have data. How do you create corresponding visual elements?
  • How do you update the visual elements if the data changes?
    • Or the user updates what they want to see…

• Scales
  • How do data values correspond to position, size, color, etc. of visual elements?

• Transitions
  • How do you smoothly animate changes between visual states?
Design considerations
Tufte’s principles of graphical excellence

• show the **data**
• induce the viewer to think about the substance rather than the methodology
• avoid distorting what the data have to say
• present **many** numbers in a small space
• make large data sets **coherent**
• encourage the eye to **compare** different pieces of data
• reveal data at several levels of detail, from overview to fine structure
• serve reasonable clear **purpose**: description, exploration, tabulation, decoration
Distortions in visualizations

• Visualizations may distort the underlying data, making it harder for reader to understand truth

• Use of **design** variation to try to falsely communicate **data** variation
Example

$(11,014)$
Example

Nobel Prizes Awarded in Science, for Selected Countries, 1901-1974

(Number of Prizes)


United States

United Kingdom

Germany

France

U.S.S.R.
Example (corrected)
Example
Traditional Electoral Map
Weighted Electoral Map
Data-ink

• Data-ink - non-redundant ink encoding data information

\[
\text{Data-ink ratio} = \frac{\text{Data-ink}}{\text{Total ink used to print the graphic}}
\]

\[= \text{proportion of a graphic's ink devoted to the non-redundant display of data-information}\]

\[= 1.0 - \text{proportion of a graphic that can be erased}\]
Examples of data-ink ratio

~0

1.0
Design principles for data-ink

• (a.k.a. aesthetics & minimalism / elegance & simplicity)

• **Above all else show the data**
  • Erase non-data-ink, within reason
    • Often not valuable and distracting
    • Redundancy not usually useful
Example
Example (revised)
**D3.js** is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3’s emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.
D3.js

- Most popular information visualization framework for the web
  - Designed by Mike Bostock as part of his PhD
- Transform data into a visual representation
  - e.g., build HTML elements for elements in an array
- Based on web standards, including HTML, CSS, SVG
Learning D3

• Many tutorials
• Many, **many** examples
  • Frequent pattern: copy similar visualization, customize for your needs
• But… be careful you use d3 v4
  • Current version
Key concepts we’ll cover today

• Selections
• Dynamic properties
• Data joins (a.k.a. data binding)
• Scales
• SVG
• Loading data
Selections

- D3 handles the mapping between data and DOM elements using "selectors"

Example: Select all <p> tags, set their color to be white

```javascript
    d3.selectAll("p").style("color", "white");
```

Example: Select all <p> tags, set their color based on some callback function:

```javascript
    d3.selectAll("p").style("color", function(data, index) {
        return index % 2 ? "black" : "gray";
    });
```
Data binding

- We can style elements dynamically based on data.
- But...
  - usually we have a dataset (e.g., time-series data of temperature readings)
  - and we want to directly associate it with some visual elements
  - and it’d be great if we could automatically create elements based on the data.
  - and delete or update the visual elements when the data changes.
Data binding

- Bind *data* with visual element.

```javascript
<code>
d3.selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .style("font-size", function(d) { return d + "px"; });
</code>
```
Data binding is persistent

- D3 uses cascade pattern, returning element set.
- By default, visual elements persist once created.
- Can update style without binding to data again.

```javascript
var p = d3.selectAll("p")
   .data([4, 8, 15, 16, 23, 42])
   .style("font-size", function(d) { return d + "px"; });

p.style("color", "blue");
```
How do we deal with changing data?
Handling Changing Data

• React:
  • Make components, bind components to state, update state

• D3:
  • Need to provide more control to rendering
  • E.g.: What if I want to highlight data that is new?
Thinking in Joins

- Elements in selection set undergo data join with elements in data

Stuff not on left  Stuff in both  Stuff not on right

https://bost.ocks.org/mike/join/
Thinking in joins

- Extra data —> enter set
- Matched data with elements —> update set
- Extra elements —> exit set
Thinking in Joins

<!- no p elements ->

var p = d3.selectAll("p")
.data([4, 8, 15, 16, 23, 42])

4, 8, 15, 16, 23, 42

.enter(...)  // update (default)  .exit(...)
Thinking in Joins

```
var p = d3.selectAll("p")
  .data([4, 8]);
```

```
<p>P1</p>
<p>P2</p>
<p>P3</p>
<p>P4</p>
```

```
<p>P1</p>  4
<p>P2</p>  8
```

```
<...> // update (default)
```

```
.exit(...)
```

```
<...>  <p>P3</p>
      <p>P4</p>
```

```
<...>  .enter(...)  <p>P1</p>
      <p>P2</p>
```

```
<...>
```
Putting it together

// Update...
var p = d3.select("body")
 .selectAll("p")
 .data([4, 8, 15, 16, 23, 42])
 .text(function(d) { return d; });

// Enter...
p.enter().append("p")
 .text(function(d) { return d; });

// Exit...
p.exit().remove();

Select all <p> in the <body>
Set the data...

When new data shows up
(not in the DOM)
Create it

When data is removed delete it

• Common pattern on data change is to rebind data to elements and separately handle
• existing elements that should have new visual style (update)
• new elements that should be created
• existing elements that should be deleted
Loading data

• What is data?
  • Anything that is an array
  • `.data()` just cares that it is an array of elements
  • Could be array of numbers, strings, JSON objects
  • If you have a dataset that is an array of JSON objects, pass it to data and you are done

```
.data([[“a”: 5}, {“a”: 3}, {“a”: 7}])
 .text(function(d) { return d.a - 1; });
```
Scaling to fit data

- 10 is a magic number
  - Transforms number in data scale to number in visual representation (“range”) scale
  - Every “1” unit in data should correspond to some unit in output coordinate system
- We’d like to automatically generate reasonable sizes, sizing data to take up all of the space based on range of data, etc.
Scales

```javascript
var x = d3.scale.linear()
    .domain([0, d3.max(data)])
    .range([0, 420]);
```

- Different types of scales that map domain values (data space) to range values (display space)
- Linear scale uses linear function (e.g., $ax + b$) to create range value from domain value

**Use:**
- Specify min and max of data
- Specify min and max of range (output)
- Generates a function (e.g., $x$) that will compute range value for a domain value
Demo: Generated SVG Bar Chart

http://jsbin.com/baqeyovaho/edit?html,js,output
Using D3

- Best place to start
  - Example code of similar visualization
  - Don’t need to understand *everything*, just enough to make it work


Visual Index
React + D3

- It’s a mess
- React and D3 both want to manipulate the DOM
- Options:
  - Let React handle the DOM (D3 can’t update it though)
  - Use React’s virtual DOM, let D3 update the virtual DOM (not very fast)
  - Ditch D3 or React :/