Data Visualization (CIS/DSC 468)

Design & D3 Introduction

Dr. David Koop
## Tasks

**What?**

1. **Analyze**
   - **Consume**
     - Discover
     - Present
     - Enjoy
   - **Produce**
     - Annotate
     - Record
     - Derive

2. **Search**
   - Target known
     - *Look up*
   - Target unknown
     - *Locate*
     - *Explore*

3. **Query**
   - Identify
   - Compare
   - Summarize

**Why?**

1. **All Data**
   - Trends
   - Outliers
   - Features

2. **Attributes**
   - One
     - Distribution
   - Many
     - Dependency
     - Correlation
     - Similarity

3. **Network Data**
   - Topology
     - Paths

4. **Spatial Data**
   - Shape

---

[Munzner (ill. Maguire), 2014]
Explore MTA Fare Data
Present Known Information

Each solid circle represents a bee species active in Carlinville, Ill., in both the late 1800s and 2010. Hatching represents a bee species active in the 1800s but now locally extinct. The spot where each block rests on the circle indicates one of 26 plant species frequented by these bees.

In the 1880s, scientists observed the following about the bee-plant encounters:
- Present
- Frequent
- Abundant

Studies in 2009 and 2010 showed many bee-plant interactions had changed:
- Lost
- Persisted
- New

[M. Stefaner, 2013]
Name Voyager

NameVoyager: Explore baby names and name trends letter by letter
Looking for the perfect baby name? Sign up for free to receive access to our expert tools!

Baby Name > An| Both  Boys  Girls  boys 100  500  100  25  1  girls 100  500  100  25  1

Names starting with 'AN' per million babies

Annotation: Circle Annotations

[S. Lu, 2017]
Record: Provenance of MTA Data Exploration

- Initial data
  - Corrected data
    - November ff
      - Sum of ffs
    - November 2 data
    - August 16 Tab
  - Station locations
    - Station map
    - Added fares
      - Difference
    - Broadway line
      - August 16
        - Broadway diff map
    - Concourse line
      - Filtered
      - Heatmap
Derived Data

Original Data

Derived Data

\[ \text{trade balance} = \text{exports} - \text{imports} \]
**Actions: Search**

<table>
<thead>
<tr>
<th></th>
<th>Target known</th>
<th>Target unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location known</td>
<td><img src="#" alt="Lookup" /></td>
<td><img src="#" alt="Browse" /></td>
</tr>
<tr>
<td>Location unknown</td>
<td><img src="#" alt="Locate" /></td>
<td><img src="#" alt="Explore" /></td>
</tr>
</tbody>
</table>

- What does a user know?
  - Lookup: check bearings
  - Locate: find on a map
  - Browse: what’s nearby
  - Explore: where to go (patterns)

[Munzner (ill. Maguire), 2014]
• Number of targets: One, Some (Often 2), or All
• Identify: characteristics or references
• Compare: similarities and differences
• Summarize: overview of everything
“Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively.”
Design Iteration

http://chartsnthings.tumblr.com/post/62679766588/19-sketches-of-quarterback-timelines
Design Iteration

<table>
<thead>
<tr>
<th>Team</th>
<th>Quarterbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York Giants</td>
<td>Eli Manning</td>
</tr>
<tr>
<td>Indianapolis Colts</td>
<td>Peyton Manning</td>
</tr>
<tr>
<td>San Diego Chargers</td>
<td>Drew Brees, Drew Brees, Philip Rivers</td>
</tr>
<tr>
<td>Baltimore Ravens</td>
<td>Kyle Boller, Steve McNair, Joe Flacco</td>
</tr>
<tr>
<td>New England Patriots</td>
<td>Tom Brady, Matt Cassel, Tom Brady</td>
</tr>
<tr>
<td>Green Bay Packers</td>
<td>Brett Favre, Aaron Rodgers, Aaron Rodgers</td>
</tr>
<tr>
<td>New Orleans Saints</td>
<td>Aaron Brooks, Drew Brees</td>
</tr>
<tr>
<td>Atlanta Falcons</td>
<td>Michael Vick, Michael Vick, Matt Ryan</td>
</tr>
<tr>
<td>New York Jets</td>
<td>Chad Pennington, Brett Favre, Mark Sanchez</td>
</tr>
<tr>
<td>Cincinnati Bengals</td>
<td>Carson Palmer, Carson Palmer</td>
</tr>
<tr>
<td>Houston Texans</td>
<td>David Carr, Matt Schaub</td>
</tr>
<tr>
<td>Carolina Panthers</td>
<td>Jake Delhomme, Jake Delhomme</td>
</tr>
<tr>
<td>Denver Broncos</td>
<td>Jake Plummer, Jay Cutler</td>
</tr>
<tr>
<td>Arizona Cardinals</td>
<td>Matt Lein, Kurt Warner</td>
</tr>
<tr>
<td>Jacksonville Jaguars</td>
<td>Byron Leftwich, David Garrard, David Garrard</td>
</tr>
<tr>
<td>Detroit Lions</td>
<td>Joey Harrington, Jon Kitna, Matthew Stafford</td>
</tr>
<tr>
<td>Tampa Bay Buccaneers</td>
<td>Chris Simms, Bruce Grac, Jeff Garcia</td>
</tr>
<tr>
<td>Dallas Cowboys</td>
<td>Drew Bledsoe, Tony Romo, Tony Romo</td>
</tr>
</tbody>
</table>
Design Iteration

Each streak shows consecutive starts by a quarterback for a single team. Streaks include playoffs.

Only two players have longer streaks: Brett Favre (275) and Eli’s brother, Peyton (227).

Among active players, Philip Rivers (122) and Joe Flacco (96) are closest behind Eli.

Find a quarterback

Eli Manning (149)

http://chartsnthings.tumblr.com/post/62679766588/19-sketches-of-quarterback-timelines
Design

• Unlike a math problem, there are many different approaches for the visualization of some data

• Need to have some way to discuss how to determine whether a visualization is doing what we want

• Validation: Understand why a design is effective
  - What problems can be effective
  - Do this at different levels

• McKenna's Slides
Four Nested Levels of Design

1. Domain situation
2. Data/task abstraction
3. Visual encoding/interaction idiom
4. Algorithm
Potential problems at each level

- **Domain situation**
  - You misunderstood their needs

- **Data/task abstraction**
  - You’re showing them the wrong thing

- **Visual encoding/interaction idiom**
  - The way you show it doesn’t work

- **Algorithm**
  - Your code is too slow
Validation at each level

- **Threat** Wrong problem
  - **Validate** Observe and interview target users

- **Threat** Wrong task/data abstraction
  - **Validate** Justify encoding/interaction design
    - **Validate** Implement system
    - **Validate** Analyze computational complexity
      - **Validate** Measure system time/memory
  - **Validate** Qualitative/quantitative result image analysis
    - *Test on any users, informal usability study*
  - **Validate** Lab study, measure human time/errors for task

- **Validate** Test on target users, collect anecdotal evidence of utility
- **Validate** Field study, document human usage of deployed system
- **Validate** Observe adoption rates
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 tools with a rich API, D3 gives you the freedom to create interactive, responsive, and aesthetically appealing data visualizations.
JavaScript Libraries

• Building Blocks: HTML, CSS, SVG, and JavaScript
• More Ideas:
  
  - JavaScript Libraries
    • `<script src="http://d3js.org/d3.v4.js" charset="utf-8"></script>`
  
  - Minification: smaller code, no functional change
    • `<script src="http://d3js.org/d3.v4.min.js" charset="utf-8"></script>`
    • Can make debugging more difficult
  
  - Content Delivery Networks
    • Faster delivery of Web content, also works for js
    • https://cdnjs.cloudflare.com/ajax/libs/d3/4.5.0/d3.min.js
JavaScript Reminders

- Functions are first-class objects in JavaScript
- Closures are functions that remember their environment
- Method Chaining: methods can also return the objects passed in or derivative objects to allow you to call another function on the result
  - You often end up following specific patterns where an object being manipulated requires multiple calls:
    - rect.attr("width", 200).attr("height", 100);
  - Or it is clear that the method returns a specific object that you wish to make changes to:
    - svg.select("#myrect").style("fill", "blue");
  - Of course, you may store the returned object as a variable and make each call separately
  - Coding style: Indent, often put each call on a new line
Data-Driven Documents (D3)

- [http://d3js.org/](http://d3js.org/)
- Original Authors: Mike Bostock, Vadim Ogievetsky, and Jeff Heer
- Open Source
- Focus on Web standards, customization, and usability
- Grew from work on Protovis: more standard, more interactive
- By nature, a **low-level** library; you have control over all elements and styles if you wish
- A top project on GitHub (over 60,000 stars as of 2/8/2017)
- Lots of impressive examples
  - Bostock was a New York Times Graphics Editor
  - [http://bost.ocks.org/mike/](http://bost.ocks.org/mike/)
D3 Key Features

• Supports data as a core piece of Web elements
  - Loading data
  - Dealing with changing data (joins, enter/update/exit)
  - **Correspondence** between data and DOM elements

• Selections (similar to CSS) that allow greater manipulation

• Method Chaining

• Integrated layout algorithms, axes calculations, etc.

• Focus on interaction support
  - Straightforward support for transitions
  - Event handling support for user-initiated changes
D3 Introduction

• Ogievetsky has put together a nice set of interactive examples that show off the major features of D3

• http://dakoop.github.io/IntroD3/
  - (Updated from original for D3 v4)

• Other references:
  - Murrany’s book on Interactive Data Visualization for the Web
  - The D3 website: d3js.org
  - Ros's Slides on v4: https://iros.github.io/d3-v4-whats-new/