Data Visualization (DSC 530/CIS 568)

Web Programming & Data

Dr. David Koop
Hyper Text Markup Language (HTML)

- Markup languages allow users to encode the **semantics** of text
- Tags define the boundaries of the structures of the content
  - Tags are enclosed in angle brackets (e.g. `<html>`)
  - Most of the time, you have a start and end tag
  - End tags are just like start tags except that they have forward slash after the open bracket (e.g. `</html>`)
  - Tags may be nested but not mismatched
    - `<p>A <strong><em>very</em></strong> cool example</p>`
    - `<p>A <strong>very <em>cool</em></strong> example</p>`
  - What about `<img src="mypicture.png" alt="My Image">`
Cascading Style Sheets (CSS)

• How do we specify what part(s) of the page we want to style?
• The **element types** themselves (the HTML tag)
  - `strong { color: red; }`
• **Classes** of elements (ties to HTML `class` attribute)
  - `.cool { color: blue; }`
• A **specific** element (ties to HTML `id` attribute)
  - `#main-section { color: green; }`
• Relationships
  - Descendant: `p em { color: yellow; }`
  - Child: `p > em { color: orange; }`
• Pseudo-classes: `a:hover { color: purple; }`
Scalable Vector Graphics (SVG)

- Vector vs. raster graphics
- Another markup language:
  - Describe the shapes and paths by their endpoints, characteristics
- SVG can be embedded into HTML5 documents!
- Pixel Coordinates: Top-left origin

(0,0)  (width,0)  

(0,0)  (width,height)
Assignment 1

- Due Monday, Feb. 11
- Questions?
SVG Grouping

- Very powerful, useful for animations and transformations
- `<g> <circle .../> <circle ... /> <circle ... /></g>`
- Can add transforms to the group:
  - http://codepen.io/dakoop/pen/rjpdXp

```xml
<svg width="200" height="200">
  <g transform="translate(0, 200) scale(1, -1)"
    <circle cx="50" cy="50" r="10"/>
  <circle cx="80" cy="80" r="10"/>
  <circle cx="110" cy="50" r="10"/>
  <circle cx="140" cy="90" r="10"/>
  </g>
</svg>
```

[SVG Example, Scheidegger, 2016]
JavaScript in one slide

- Interpreted and Dynamically-typed Programming Language
- Statements end with semi-colons, normal blocking with brackets
- Variables: `var a = 0; let b = 2;`
- Operators: `+, -, *, /, [], {}`
- Control Statements: `if (<expr>) {...} else {...}`, `switch`
- Loops: `for, while, do-while`
- Arrays: `var a = [1,2,3]; a[99] = 100; console.log(a.length);`
- Functions: `function myFunction(a,b) { return a + b; }`
- Objects: `var obj; obj.x = 3; obj.y = 5;`
  - Prototypes for instance functions
- Comments are `/* Comment */` or `// Single-line Comment`
JavaScript Objects

- var student = {name: "John Smith", id: "000012345", class: "Senior", hometown: "Fall River, MA, USA"};

- Objects contain multiple values: key-value pairs called **properties**

- Accessing properties via dot-notation: student.name

- May also contain functions:
  - var student =
    {firstName: "John",
     lastName: "Smith",
     fullName: function() { return this.firstName + " " + this.lastName; }};
  - student.fullName()

- JavaScript Object Notation (JSON): data interchange format
  - nested objects and arrays (data only, no functions!)  
  - **subset** of JavaScript
Objects as Associative Arrays/Dictionaries

• Objects have key-value pairs and can be addressed via those keys, either via dot-notation or via bracket notation: [<key>]

• Example:

```javascript
states = {"AZ": "Arizona", "MA": "Massachusetts", ...

  // Get a state's name given it's abbreviation
  console.log("MA is" + states["MA"])"
```

• Similar to dictionaries or associative arrays in other languages (e.g. Python)

• Dot-notation only works with certain identifiers, bracket notation works with more identifiers
Functional Programming in JavaScript

• Functions are first-class objects in JavaScript
• You can pass a function to a method just like you can pass an integer, string, or object
• Instead of writing loops to process data, we can instead use a map/filter/reduce/forEach function on the data that will run our logic for each data item.
• **map**: transform each element of an array
• **filter**: check each element of an array and keep only ones that pass
• **forEach**: run the function for each element of the array
• **reduce**: collapse an array to a single object
Quiz

• Given this data:
  - `var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];`

• Questions:
  - How would I return a new array with values one less than `a`?
  - How would I find only the values `>= 10`?
  - How would I sum the array?
  - How would I create a reversed version of the array?
Quiz Answers: Notebook

• Data: \( \text{var } a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6]; \)
• How would I subtract one from each item?
  - \( a.\text{map}(\text{function}(d) \{ \text{return } d-1; \}) \)
• How would I find only the values \( \geq 10? \)
  - \( a.\text{filter}(\text{function}(d) \{ \text{return } d \geq 10; \}) \)
• How would I sum the array?
  - \( a.\text{reduce}(\text{function}(s,d) \{ \text{return } s + d; \}) \)
• How would I create a reversed version of the array?
  - \( b = []; \)
    - \( a.\text{forEach}(\text{function}(d) \{ b.\text{unshift}(d); \}); \)
    - ...or \( a.\text{reverse}() \) // modifies in place
• Arrow functions shorten such calls: \( a.\text{map}(d \rightarrow d-1); \)
  - \( a.\text{filter}(d \rightarrow d \geq 10); \) \( a.\text{reduce}((s,d) \rightarrow s+d); \)
Function Chaining in JavaScript

- When programming functionally, it is useful to chain functions.
- No intermediate variables!
- Often more readable code.
- jQuery Example:
  - `$('#myElt').css('color', 'blue').height(200).width(320)`
- Used a lot in Web programming, especially D3.
- Can return the same object or a new object.
- Lazy chaining keeps track of functions to be applied but will apply them later (e.g. when the page loads).
Closures in JavaScript

- Functions can return functions with some values set
- Allows assignment of some of the values
- Closures are functions that "remember their environments" [MDN]
- Example (MDN):

  ```javascript
  function makeAdder(x) {
    return function(y) {
      return x + y;
    };
  }
  var add5 = makeAdder(5);
  var add10 = makeAdder(10);

  console.log(add5(2)); // 7
  console.log(add10(2)); // 12
  ```

- Notebook
Manipulating the DOM with JavaScript

• Key global variables:
  • `window`: Global namespace
  • `document`: Current document
  • `document.getElementById(...)`: Get an element via its id

• HTML is parsed into an in-memory document (DOM)
• Can access and **modify** information stored in the DOM
• Can add information to the DOM
Example: JavaScript and the DOM

• Start with no real content, just divs:
  <div id="firstSection"></div>
  <div id="secondSection"></div>
  <div id="finalSection"></div>

• Get existing elements:
  - document.querySelector
  - document.getElementById

• Programmatically add elements:
  - document.createElement
  - document.createTextNode
  - Element.appendChild
  - Element.setAttribute

• Link
Creating SVG figures via JavaScript

• SVG elements can be accessed and modified just like HTML elements

• Create a new SVG programmatically and add it into a page:
  
  ```javascript
  var divElt = document.getElementById("chart");
  var svg = document.createElementNS(
    "http://www.w3.org/2000/svg", "svg");
  divElt.appendChild(svg);
  ```

• You can assign attributes:
  
  ```javascript
  svg.setAttribute("height", 400);
  svg.setAttribute("width", 600);
  svgCircle.setAttribute("r", 50);
  ```
Manipulating SVG via JavaScript

- SVG can be navigated just like the DOM
- Example:

  ```javascript
  function addEltToSVG(svg, name, attrs) {
    var element = document.createElementNS("http://www.w3.org/2000/svg", name);
    if (attrs === undefined) attrs = {};
    for (var key in attrs) {
      element.setAttribute(key, attrs[key]);
    }
    svg.appendChild(element);
  }

  mysvg = document.getElementById("mysvg");
  addEltToSVG(mysvg, "rect", {"x": 50, "y": 50,
    "width": 40,"height": 40,
    "fill": "blue"});
  ```

- Notebook
SVG Manipulation Example

- Draw a horizontal bar chart
  
  ```javascript
  var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];
  ```

- Steps?
SVG Manipulation Example

• Draw a horizontal bar chart
  - var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];

• Steps:
  - Programmatically create SVG
  - Create individual rectangle for each item

• Link:
  - https://codepen.io/dakoop/pen/pabrRa
JavaScript References

- Interactive Data Visualization for the Web, Murray
- MDN Tutorials
“Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively.”

— T. Munzner
Data

• What is this data?

<p>| | | | | | | | | |</p>
<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>R011</td>
<td>42ND STREET &amp; 8TH AVENUE</td>
<td>00228985</td>
<td>00008471</td>
<td>00000441</td>
<td>00001455</td>
<td>00000134</td>
<td>00033341</td>
<td>00071255</td>
</tr>
<tr>
<td>R170</td>
<td>14TH STREET-UNION SQUARE</td>
<td>00224603</td>
<td>00011051</td>
<td>00000827</td>
<td>00003026</td>
<td>00000660</td>
<td>00089367</td>
<td>00199841</td>
</tr>
<tr>
<td>R046</td>
<td>42ND STREET &amp; GRAND CENTRAL</td>
<td>00207758</td>
<td>00007908</td>
<td>00000323</td>
<td>00001183</td>
<td>00003001</td>
<td>00040759</td>
<td>00096613</td>
</tr>
</tbody>
</table>

• **Semantics**: real-world meaning of the data
• **Type**: structural or mathematical interpretation
• Both often require **metadata**
  - Sometimes we can infer some of this information
  - Line between data and metadata isn’t always clear
## Data

<table>
<thead>
<tr>
<th>REMOTE</th>
<th>STATION</th>
<th>FF</th>
<th>SEN/DIS</th>
<th>7-D AFAS UNL</th>
<th>D AFAS/RMF</th>
<th>JOINT RR TKT</th>
<th>7-D UNL</th>
<th>30-D UNL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R011 42ND STREET &amp; 8TH AVENUE</td>
<td>00228985</td>
<td>00008471</td>
<td>00000441</td>
<td>00001455</td>
<td>00000134</td>
<td>0033341</td>
<td>0071255</td>
</tr>
<tr>
<td>2</td>
<td>R170 14TH STREET-UNION SQUARE</td>
<td>00224603</td>
<td>0011051</td>
<td>00000827</td>
<td>00003026</td>
<td>00000660</td>
<td>00089367</td>
<td>00199841</td>
</tr>
<tr>
<td>3</td>
<td>R046 42ND STREET &amp; GRAND CENTRAL</td>
<td>00207758</td>
<td>0007908</td>
<td>0000323</td>
<td>00001183</td>
<td>00003001</td>
<td>0040759</td>
<td>0096613</td>
</tr>
<tr>
<td>4</td>
<td>R012 34TH STREET &amp; 8TH AVENUE</td>
<td>00188311</td>
<td>0006490</td>
<td>00000498</td>
<td>00001279</td>
<td>00003622</td>
<td>0035527</td>
<td>0067483</td>
</tr>
<tr>
<td>5</td>
<td>R293 34TH STREET - PENN STATION</td>
<td>00168768</td>
<td>0006155</td>
<td>0000523</td>
<td>00001065</td>
<td>00005031</td>
<td>0030645</td>
<td>0054376</td>
</tr>
<tr>
<td>6</td>
<td>R033 42ND STREET/TIMES SQUARE</td>
<td>00159382</td>
<td>0005945</td>
<td>0000378</td>
<td>00001205</td>
<td>0000690</td>
<td>0058931</td>
<td>0078644</td>
</tr>
<tr>
<td>7</td>
<td>R022 34TH STREET &amp; 6TH AVENUE</td>
<td>00156008</td>
<td>0006276</td>
<td>0000487</td>
<td>00001543</td>
<td>0000712</td>
<td>0058910</td>
<td>0110466</td>
</tr>
<tr>
<td>8</td>
<td>R084 59TH STREET/COLUMBUS CIRCLE</td>
<td>00155262</td>
<td>0009484</td>
<td>0000589</td>
<td>0002071</td>
<td>0000542</td>
<td>0053397</td>
<td>0113966</td>
</tr>
<tr>
<td>9</td>
<td>R020 47-50 STREETS/ROCKEFELLER</td>
<td>00143500</td>
<td>0006402</td>
<td>0000384</td>
<td>0001159</td>
<td>0000723</td>
<td>0037978</td>
<td>0090745</td>
</tr>
<tr>
<td>10</td>
<td>R179 86TH STREET-LEXINGTON AVE</td>
<td>00142169</td>
<td>0010367</td>
<td>0000470</td>
<td>0001839</td>
<td>0000271</td>
<td>0050328</td>
<td>0125250</td>
</tr>
<tr>
<td>11</td>
<td>R023 34TH STREET &amp; 6TH AVENUE</td>
<td>00134052</td>
<td>0005005</td>
<td>0000348</td>
<td>0001112</td>
<td>0000649</td>
<td>0031531</td>
<td>0075040</td>
</tr>
<tr>
<td>12</td>
<td>R029 PARK PLACE</td>
<td>00121614</td>
<td>0004311</td>
<td>0000287</td>
<td>0000931</td>
<td>0000792</td>
<td>0025404</td>
<td>0065362</td>
</tr>
<tr>
<td>13</td>
<td>R047 42ND STREET &amp; GRAND CENTRAL</td>
<td>00100742</td>
<td>0004273</td>
<td>0000185</td>
<td>0000704</td>
<td>0001241</td>
<td>0022808</td>
<td>0068216</td>
</tr>
<tr>
<td>14</td>
<td>R031 34TH STREET &amp; 7TH AVENUE</td>
<td>00095076</td>
<td>0003990</td>
<td>0000232</td>
<td>0000727</td>
<td>0001459</td>
<td>0024284</td>
<td>0038671</td>
</tr>
<tr>
<td>15</td>
<td>R017 LEXINGTON AVENUE</td>
<td>00094655</td>
<td>0004688</td>
<td>0000190</td>
<td>0000833</td>
<td>0000754</td>
<td>0020018</td>
<td>0055066</td>
</tr>
<tr>
<td>16</td>
<td>R175 8TH AVENUE-14TH STREET</td>
<td>00094313</td>
<td>0003907</td>
<td>0000286</td>
<td>0001144</td>
<td>0000256</td>
<td>0038272</td>
<td>0074661</td>
</tr>
<tr>
<td>17</td>
<td>R057 BARCLAYS CENTER</td>
<td>00093804</td>
<td>0004204</td>
<td>0000454</td>
<td>0001386</td>
<td>0001491</td>
<td>0039113</td>
<td>0068119</td>
</tr>
<tr>
<td>18</td>
<td>R138 WEST 4TH ST-WASHINGTON SQ</td>
<td>00093562</td>
<td>0004677</td>
<td>0000251</td>
<td>0000965</td>
<td>0000127</td>
<td>0031628</td>
<td>0074458</td>
</tr>
</tbody>
</table>
Data Terminology

• Items
  - An item is an individual discrete entity
  - e.g. row in a table, node in a network

• Attributes
  - An attribute is some specific property that can be measured, observed, or logged
    - a.k.a. variable, (data) dimension
  - e.g. a column in a table
Semantics

• The meaning of the data
• Example: 94023, 90210, 52790, 02747
Semantics

• The meaning of the data
• Example: 94023, 90210, 52790, 02747
  - Attendance at college football games?
Semantics

• The meaning of the data
• Example: 94023, 90210, 52790, 02747
  - Attendance at college football games?
  - Salaries?
Semantics

• The meaning of the data

• Example: 94023, 90210, 52790, 02747
  - Attendance at college football games?
  - Salaries?
  - Zip codes?

• Cannot always infer based on what the data looks like

• Often require semantics to better understand data

• Column names help with semantics

• May also include rules about data: a zip code is part of an address that uniquely identifies a residence

• Useful for asking good questions about the data
## Items & Attributes

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
<th>T</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Order ID</td>
<td>Order Date</td>
<td>Order Priority</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>10/14/06</td>
<td>5-Low</td>
<td>Large Box</td>
<td>10/21/06</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>2/21/08</td>
<td>4-Not Specified</td>
<td>Small Pack</td>
<td>2/22/08</td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>7/16/07</td>
<td>2-High</td>
<td>Small Pack</td>
<td>7/17/07</td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>7/16/07</td>
<td>2-High</td>
<td>Jumbo Box</td>
<td>7/17/07</td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>7/16/07</td>
<td>2-High</td>
<td>Medium Box</td>
<td>7/18/07</td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>7/16/07</td>
<td>2-High</td>
<td>Medium Box</td>
<td>7/18/07</td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>10/23/07</td>
<td>4-Not Specified</td>
<td>Wrap Bag</td>
<td>10/24/07</td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>10/23/07</td>
<td>4-Not Specified</td>
<td>Small Box</td>
<td>10/25/07</td>
</tr>
<tr>
<td>36</td>
<td>3</td>
<td>11/3/07</td>
<td>1-Urgent</td>
<td>Small Box</td>
<td>11/3/07</td>
</tr>
<tr>
<td>65</td>
<td>3</td>
<td>3/18/07</td>
<td>1-Urgent</td>
<td>Small Pack</td>
<td>3/19/07</td>
</tr>
<tr>
<td>66</td>
<td>3</td>
<td>1/30/05</td>
<td>5-Low</td>
<td>Wrap Bag</td>
<td>1/20/05</td>
</tr>
<tr>
<td>69</td>
<td>3</td>
<td>5/4-Not Specified</td>
<td>Small Pack</td>
<td>6/6/05</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>3</td>
<td>5/4-Not Specified</td>
<td>Wrap Bag</td>
<td>6/6/05</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>3</td>
<td>12/18/06</td>
<td>5-Low</td>
<td>Small Box</td>
<td>12/23/06</td>
</tr>
<tr>
<td>70</td>
<td>3</td>
<td>12/18/06</td>
<td>5-Low</td>
<td>Wrap Bag</td>
<td>12/23/06</td>
</tr>
<tr>
<td>96</td>
<td>3</td>
<td>4/17/05</td>
<td>2-High</td>
<td>Small Box</td>
<td>4/19/05</td>
</tr>
<tr>
<td>97</td>
<td>3</td>
<td>1/29/06</td>
<td>3-Medium</td>
<td>Small Box</td>
<td>1/30/06</td>
</tr>
<tr>
<td>129</td>
<td>3</td>
<td>11/19/08</td>
<td>5-Low</td>
<td>Small Box</td>
<td>11/28/08</td>
</tr>
<tr>
<td>130</td>
<td>3</td>
<td>5/8/08</td>
<td>2-High</td>
<td>Small Box</td>
<td>5/9/08</td>
</tr>
<tr>
<td>130</td>
<td>3</td>
<td>5/8/08</td>
<td>2-High</td>
<td>Medium Box</td>
<td>5/10/08</td>
</tr>
<tr>
<td>130</td>
<td>3</td>
<td>5/8/08</td>
<td>2-High</td>
<td>Small Box</td>
<td>5/11/08</td>
</tr>
<tr>
<td>132</td>
<td>3</td>
<td>6/11/06</td>
<td>3-Medium</td>
<td>Medium Box</td>
<td>6/12/06</td>
</tr>
<tr>
<td>132</td>
<td>3</td>
<td>6/11/06</td>
<td>3-Medium</td>
<td>Jumbo Box</td>
<td>6/14/06</td>
</tr>
<tr>
<td>134</td>
<td>3</td>
<td>5/1/08</td>
<td>4-Not Specified</td>
<td>Large Box</td>
<td>5/3/08</td>
</tr>
<tr>
<td>135</td>
<td>3</td>
<td>10/21/07</td>
<td>4-Not Specified</td>
<td>Small Pack</td>
<td>10/23/07</td>
</tr>
<tr>
<td>166</td>
<td>3</td>
<td>9/12/07</td>
<td>2-High</td>
<td>Small Box</td>
<td>9/14/07</td>
</tr>
<tr>
<td>193</td>
<td>3</td>
<td>8/8/06</td>
<td>1-Urgent</td>
<td>Medium Box</td>
<td>8/10/06</td>
</tr>
<tr>
<td>194</td>
<td>3</td>
<td>4/5/08</td>
<td>3-Medium</td>
<td>Wrap Bag</td>
<td>4/7/08</td>
</tr>
</tbody>
</table>

**Attributes**
- **Order ID**: Unique identifier for each order.
- **Order Date**: Date the order was placed.
- **Order Priority**: Priority level assigned to the order.
- **Product Container**: Type of container used to ship the order.
- **Product Base Margin**: Base margin for the product.
- **Ship Date**: Date the order was shipped.
Data Types

- **Nodes**
  - Synonym for item but in the context of networks (graphs)

- **Links**
  - A **link** is a relation between two items
  - e.g. social network friends, computer network links
Items & Links

[Bostock, 2011]