Data Visualization (CIS 468)

Web Programming

Dr. David Koop
Definition of Visualization

“Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively.”

— T. Munzner
Why Visual?

<table>
<thead>
<tr>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>10.0</td>
<td>8.04</td>
<td>10.0</td>
<td>9.14</td>
</tr>
<tr>
<td>8.0</td>
<td>6.95</td>
<td>8.0</td>
<td>8.14</td>
</tr>
<tr>
<td>13.0</td>
<td>7.58</td>
<td>13.0</td>
<td>8.74</td>
</tr>
<tr>
<td>9.0</td>
<td>8.81</td>
<td>9.0</td>
<td>8.77</td>
</tr>
<tr>
<td>11.0</td>
<td>8.33</td>
<td>11.0</td>
<td>9.26</td>
</tr>
<tr>
<td>14.0</td>
<td>9.96</td>
<td>14.0</td>
<td>8.10</td>
</tr>
<tr>
<td>6.0</td>
<td>7.24</td>
<td>6.0</td>
<td>6.13</td>
</tr>
<tr>
<td>4.0</td>
<td>4.26</td>
<td>4.0</td>
<td>3.10</td>
</tr>
<tr>
<td>12.0</td>
<td>10.84</td>
<td>12.0</td>
<td>9.13</td>
</tr>
<tr>
<td>7.0</td>
<td>4.82</td>
<td>7.0</td>
<td>7.26</td>
</tr>
<tr>
<td>5.0</td>
<td>5.68</td>
<td>5.0</td>
<td>4.74</td>
</tr>
</tbody>
</table>

[F. J. Anscombe]
Why Visual?

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>10.0</td>
<td>8.04</td>
<td>10.0</td>
<td>9.14</td>
<td>10.0</td>
</tr>
<tr>
<td>8.0</td>
<td>6.95</td>
<td>8.0</td>
<td>8.14</td>
<td>8.0</td>
</tr>
<tr>
<td>13.0</td>
<td>7.58</td>
<td>13.0</td>
<td>8.74</td>
<td>13.0</td>
</tr>
<tr>
<td>9.0</td>
<td>8.81</td>
<td>9.0</td>
<td>8.77</td>
<td>9.0</td>
</tr>
<tr>
<td>11.0</td>
<td>8.33</td>
<td>11.0</td>
<td>9.26</td>
<td>11.0</td>
</tr>
<tr>
<td>14.0</td>
<td>9.96</td>
<td>14.0</td>
<td>8.10</td>
<td>14.0</td>
</tr>
<tr>
<td>6.0</td>
<td>7.24</td>
<td>6.0</td>
<td>6.13</td>
<td>6.0</td>
</tr>
<tr>
<td>4.0</td>
<td>4.26</td>
<td>4.0</td>
<td>3.10</td>
<td>4.0</td>
</tr>
<tr>
<td>12.0</td>
<td>10.84</td>
<td>12.0</td>
<td>9.13</td>
<td>12.0</td>
</tr>
<tr>
<td>7.0</td>
<td>4.82</td>
<td>7.0</td>
<td>7.26</td>
<td>7.0</td>
</tr>
<tr>
<td>5.0</td>
<td>5.68</td>
<td>5.0</td>
<td>4.74</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Mean of x: 9
Variance of x: 11
Mean of y: 7.50
Variance of y: 4.122
Correlation: 0.816

[F. J. Anscombe]
Why Visual?

[F. J. Anscombe]
Visual Pop-out
Visual Perception Limitations
Languages of the Web

• HTML
• CSS
• SVG
• JavaScript
  - Versions of Javascript: ES6, ES2015, ES2017…
  - Specific frameworks: react, jQuery, bootstrap, D3
HTML Exercise

• What does this HTML mean?

  - <em>This is <strong>cool</strong>. What about <u><strong>this?</strong></u></em>
Registration
Assignment 1

• Out soon
• HTML, CSS, SVG
Cascading Style Sheets (CSS)

• Separate from content, just specifies how to style the content
• Style information can appear in three places:
  - External file
  - In a style element at the beginning of the HTML file
  - In a specific element in the body of a document (least preferable)
• Why Cascading?
  - Don’t want to have to specify everything over and over
  - Often want to use the same characteristics in a region of the DOM
  - Use inheritance: properties that apply to children cascade down
CSS Selectors

• 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; }`
Other CSS Bits

- Comments: /* This is a comment in CSS */
- Grouping Selectors: `p, li { font-size: 12pt; }
- Multiple Classes: `.cool.temp { color: blue; }
- Colors:
  - Names (Level 1, 2, & 3): red, orange, antiquewhite
  - Dash notation (3- & 6-character): #fff, #00ff00
  - Integer or % RGB and HSL Functions: `rgb(255, 0, 0), rgb(50%, 50%, 0%), hsl(120, 100%, 50%)`
  - Also `background-color`
- Watch out for multiple rules (look at how a web browser parses)
- Again, much more documentation at MDN
Sample CSS

body {
  font-face: sans-serif;
  font-size: 12pt;
}

em { color: green; }

em u { color: red; }

em > strong { color: blue; }

img { border: 4px solid red; }

- What colors are displayed for this HTML (with the above stylesheet)?
  - <em>This is <strong>cool</strong>. What about <u><strong>this?</strong></u></em>
CSS Specificity

• Example:
  - CSS:
    
    p.highlight { color: red; }
    p { color: blue; }

    - What is the color of the paragraph
    
    <p class="highlight">Cool</p>?

• Generally, last rule listed overrides previous rules
• …but anytime a selector is more specific, it has precedence
• p.highlight is a more specific selector
• When in doubt, test it in a browser
How to add CSS to HTML

• External: a separate file via a link element (in the `<head>` section):
  - `<link rel="stylesheet" href="styles.css">`

• Embedded: in the header:
  - `<style type="text/css"> ... </style>`

• Inline: for a specific element: *(Discouraged!)*
  - `<p style="font-weight: bold;">Some text</p>`
CSS Exercise

body {
    font-family: sans-serif;
    font-size: 12pt;
}

div {
}

em {
    color: green;
}

em u {
    color: red;
}

em > strong {
    color: blue;
}

img {
    border: 4px solid red;
}

• What colors are displayed for this HTML (with the above stylesheet)?
   - <em>This is <strong>cool</strong>. What about <u><strong>this?</strong></u></em>
What is the difference between vector and raster graphics?
Scalable Vector Graphics (SVG)

- Vector graphics vs. Raster graphics
- Drawing commands versus a grid of pixels
- Why vector graphics?
SVG Background

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

![Pixel Coordinates Diagram]
SVG Elements

• Drawing primitives:
  - Lines, Circles, Rects, Ellipses, Text, Polylines, Paths
  - Work by specifying information about how to draw the shape
  - Lots more: see MDN Documentation

• Ordering/Stacking:
  - SVG Elements are drawn in the order they are specified

• Paths: directions for drawing
SVG Example

- [SVG Code](http://codepen.io/dakoop/pen/yexVXb)

```xml
<svg id="mysvg" width="300" height="600">
  <circle cx="50" cy="50" r="50"/>
  <rect class="lego" x="150" y="150" width="50" height="20"/>
  <path id="triangle" d="M 20 200 L 120 200 L 120 250 Z"/>
</svg>
```

- `circle` { fill: green; stroke: black; stroke-width: 4px; }
  `.lego` { fill: red; stroke: red; stroke-width: 4px; }
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](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 Styles

• Can specify styles via CSS, too

... 

```
<style type="text/css">
circle { fill: green; stroke: black; 
  stroke-width: 4px; }
.normal { fill: red; stroke: blue; 
  stroke-width: 2px; }
#p1 { fill: none; stroke: red; stroke-width: 3px; }
</style>
```

...

```
<svg id="mysvg" width="400" height="300">
  <circle cx="50" cy="50" r="50"/>
  <rect class="normal" x="150" y="150" width="50" 
    height="20"/>
  <path id="p1" d="M 200 10 L 300 10 L 300 50 Z"/>
</svg>
```

...
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`