Data Visualization (CSCI 490/680)

Data & Isosurfacing

Dr. David Koop
Focus+Content Overview

- Embed
  - Elide Data
  - Superimpose Layer
  - Distort Geometry

Reduce

- Filter
- Aggregate
- Embed

[Munzner (ill. Maguire), 2014]
Elision & Degree of Interest Function

- DOI = I(x) - D(x,y)
  - I: interest function
  - D: distance (semantic or spatial)
  - x: location of item
  - y: current focus point
  - Interactive: y changes

[Heer and Card, 2004]
Superimposition with Interactive Lenses

(a) Alteration

(b) Suppression

[ChronoLenses and Sampling Lens in Tominski et al., 2014]
It can be difficult to observe micro and macro features simultaneously with complex graphs. If you zoom in for detail, the graph is too big to view in its entirety. If you zoom out to see the overall structure, small details are lost.

Focus + context techniques allow interactive exploration of an area.
Distortion: Stretch and Squish Navigation

Figure 3. LiveRAC shows a full day of system management time-series data using a reorderable matrix of area-aware charts. Over 4000 devices are shown in rows, with 11 columns representing groups of monitored parameters. (a): The user has sorted by the maximum value in the CPU column. The first several dozen rows have been stretched to show sparklines for the devices, with the top 13 enlarged enough to display text labels. The time period of business hours has been selected, showing the increase in the In pkts parameter for many devices. (b): The top three rows have been further enlarged to show fully detailed charts in the CPU column and partially detailed ones in Swap and two other columns. The time marker (vertical black line on each chart) indicates the start of anomalous activity in several of spire’s parameters. Below the labeled rows, we see many blocks at the lowest semantic zoom level, and further below we see a compressed region of highly saturated blocks that aggregate information from many charts.

Principle: multiple views are most effective when coordinated through explicit linking.

The principle of linked views [15] is that explicit coordination between views enhances their value. In LiveRAC, as the user moves the cursor within a chart, the same point in time is marked in all charts with a vertical line. Similarly, selecting a time segment in one chart shows a mark in all of them. This technique allows direct comparison between parameter values at the same time on different charts. In addition, people can easily correlate times between large charts with detailed axis labels, and smaller, more concise charts.

Assertion: showing several levels of detail simultaneously provides useful high information density in context.

Several technique choices are based on this assertion. First, LiveRAC uses stretch and squish navigation, where expanding one or many regions compresses the rest of the view [11, 17]. The accompanying video shows the look and feel of this navigation technique. The stretching and squishing operates on rectangular regions, so expanding a single chart also magnifies the entire row for the device it represents, and the entire column for the parameters that it shows. The edges of the display are fixed so that all cells remain within the visible area, as opposed to conventional zooming where some regions are pushed off-screen. There are rapid navigation shortcuts to zoom a single cell, a column, an aggregated group of devices, the results of a search, or to zoom out to an overview. Users can also directly drag grid lines or resize freely drawn on-screen rectangles. Navigation shortcuts can also be created for any arbitrary grouping, whose cells do not need to be contiguous. This interaction mechanism affords multiple focus regions, supporting multiple levels of detail.

Second, charts in LiveRAC dynamically adapt to show visual representations adapted in each cell to the available screen space. This technique, called semantic zooming [13], allows a hierarchy of representations for a group of device-parameter time-series. In Figure 3, the largest charts have multiple overlaid curves and detailed axis and legend labels. Smaller charts show fewer curves and less labeling, and at smaller sizes only one curve is shown as a sparkline [24]. On each curve, the maximum value over the displayed time period is indicated with a red dot, the minimum with a blue dot, and the current value with a green one. All representation levels color code the background rectangle according to dynamically changeable thresholds of the minimum, maximum, or average values of the parameters within the current time window. The smallest view is a simple block, where this color coding is the only information shown.

Third, aggregation techniques achieve visual scalability by ensuring dense regions show meaningful visual representations. Given our target scale of dozens of parameters and thousands of devices, the size of the matrix could easily surpass 100,000 cells. Stretch and squish navigation allows users to quickly create a mosaic with cells of many different sizes. Stretching a single chart also magnifies the entire row for the device it represents, and the entire column for the parameters that it shows. The edges of the display are fixed so that all cells remain within the visible area, as opposed to conventional zooming where some regions are pushed off-screen. There are rapid navigation shortcuts to zoom a single cell, a column, an aggregated group of devices, the results of a search, or to zoom out to an overview. Users can also directly drag grid lines or resize freely drawn on-screen rectangles. Navigation shortcuts can also be created for any arbitrary grouping, whose cells do not need to be contiguous. This interaction mechanism affords multiple focus regions, supporting multiple levels of detail.

[McLachlan et al., 2008]
Focus+Context in Network Exploration

(a) Bring (step 1) – Selecting a node fades out (b) Bring (step 2) – Neighbor nodes are pulled (c) Go – After selecting a neighbor (the green node in Fig. 4(b)), a short animation brings the focus towards a new neighborhood.

[Lambert et al., 2010]
Distortion Concerns

• Distance and length judgments are **harder**
  - Example: Mac OS X Dock with Magnification
  - Spatial position of items changes as the focus changes
• Node-link diagrams not an issue… why?
• Users have to be made aware of distortion
  - Back to scatterplot with distortion example
  - Lenses or shading give clues to users
• **Object constancy**: understanding when two views show the same object
  - What happens under distortion?
  - 3D Perspective is distortion… but we are well-trained for that
• Think about **what** is being shown (filtering) and method (fisheye)
Designs Feedback

- Some good prototypes and focus on interactions
- Generally, would like to see more creativity
  - You can create scatterplots and choropleth maps using Tableau
  - https://xeno.graphics
  - https://www.informationisbeautifulawards.com/showcase
- Justify the use of widgets and/or tooltips
- Provide complete overviews, even if interactions will filter views or provide details
- Be careful with scrolling
Assignment 5

- Multiple Views and Interaction using Linked Highlighting
- Due November 22
Data Wrangling

- Problem 1: Visualizations need data
  - Solution: The Web!

- Problem 2: Data has extra information I don't need
  - Solution: Filter it

- Problem 3: Data is dirty
  - Solution: Clean it up

- Problem 4: Data isn't in the same place
  - Solution: Combine data from different sources

- Problem 5: Data isn't structured correctly
  - Solution: Reorder, map, and nest it
Hosting data

- github.com
- gist.github.com
- figshare.com
- myjson.com
- Other services
Why JavaScript?

• Python and R have great support for this sort of processing
• Data comes from the Web, want to put visualizations on the Web
• Sometimes unnecessary to download, process, and upload!
• More tools are helping JavaScript become a better language
JavaScript Data Wrangling Resources

- https://observablehq.com/@dakoop/learn-js-data
- Based on http://learnjsdata.com/
- Good coverage of data wrangling using JavaScript
Comma Separated Values (CSV)

• File structure:

  cities.csv:

city, state, population, land area
seattle, WA, 652405, 83.9
new york, NY, 8405837, 302.6
boston, MA, 645966, 48.3
kansas city, MO, 467007, 315.0

• Loading using D3:

  d3.csv("/data/cities.csv").then(function(data) {
    console.log(data[0]);
  });

• Result:

  => {city: "seattle", state: "WA", population: 652405, land area: 83.9}

• Values are strings! Convert to numbers via the unary + operator:

  - d.population => "652405"
  - +d.population => 652405

[http://learnjsdata.com]
Tab Separated Values (TSV)

• File structure:

```plaintext
animals.tsv:

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>avg_weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>tiger</td>
<td>mammal</td>
<td>260</td>
</tr>
<tr>
<td>hippo</td>
<td>mammal</td>
<td>3400</td>
</tr>
<tr>
<td>komodo dragon</td>
<td>reptile</td>
<td>150</td>
</tr>
</tbody>
</table>
```

• Loading using D3:

```javascript
d3.tsv("/data/animals.tsv").then(function(data) {
    console.log(data[0]);
});
```

• Result:

```javascript
=> {name: "tiger", type: "mammal", avg_weight: "260"}
```

• Can also have other delimiters (e.g. '|', ';')
JavaScript Object Notation (JSON)

• File Structure:

```javascript
employees.json:
[
  {"name":"Andy Hunt",
   "title":"Big Boss",
   "age": 68,
   "bonus": true
  },
  {"name":"Charles Mack",
   "title":"Jr Dev",
   "age":24,
   "bonus": false
  }
]
```

• Loading using D3:

```javascript
d3.json("/data/employees.json").then(function(data) {
  console.log(data[0]);
});
```

• Result:

`=> {name: "Andy Hunt", title: "Big Boss", age: 68, bonus: true}`
Loading Multiple Files

- Use Promise.all to load multiple files and then process them all

```
Promise.all([d3.csv("/data/cities.csv"),
             d3.tsv("/data/animals.tsv")])
  .then(analyze);

function analyze(data) {
  cities = data[0]; animals = data[1];
  console.log(cities[0]);
  console.log(animals[0]);
}
```

```javascript
=> {city: "seattle", state: "WA", population: "652405", land area: "83.9"}
{name: "tiger", type: "mammal", avg_weight: "260"}
```
Combining Data

- Suppose given products and brands
- Brands have an id and products have a brand_id that matches a brand
- Want to join these two datasets together
  - Product.brand_id => Brand.id
- Use a nested forEach/filter
- Use a native join command
Summarizing Data

• d3 has min, max, and extent functions of the form
  - 1st argument: dataset
  - 2nd argument: accessor function

• Example:
  ```javascript
  var landExtent = d3.extent(data, function(d) {
    return d.land_area;
  });
  console.log(landExtent); => [48.3, 315]
  ```

• Summary statistics, e.g. mean, median, deviation → same format

• Median Example:
  ```javascript
  var landMed = d3.median(data, function(d) {
    return d.land_area;
  });
  console.log(landMed);
  => 193.25
  ```
Nesting Data

• Take a flat structure and turn it into something nested
• Often similar to a groupby in databases
  - key indicate groupings
  - rollup indicates how the groups are processed/aggregated
• Last function specifies the data and how the output should look
  - entries: [{key: <key>, value: <value>}]
  - object: {<key>: <value>, ...}
  - map: {<key>: <value>, ...} but as a d3.map (safer than object, but uses get/set instead of square brackets ([]))
Nesting Example

• Data

```javascript
var expenses = [{"name":"jim","amount":34,"date":"11/12/2015"},
    {"name":"carl","amount":120.11,"date":"11/12/2015"},
    {"name":"jim","amount":45,"date":"12/01/2015"},
    {"name":"stacy","amount":12.00,"date":"01/04/2016"},
    {"name":"stacy","amount":34.10,"date":"01/04/2016"},
    {"name":"stacy","amount":44.80,"date":"01/05/2016"}];
```

• Using d3.nest:

```javascript
var expensesAvgAmount = d3.nest()
    .key(function(d) { return d.name; })
    .rollup(function(v) { return d3.mean(v, function(d) { return d.amount; }); });
entries(expenses);
console.log(JSON.stringify(expensesAvgAmount));
```

• Result:

```javascript
=> [{"key":"jim","values":39.5},
    {"key":"carl","values":120.11},
    {"key":"stacy","values":30.3}]
```
d3-array 2.0 Updates

- Works with iterables
- group and rollup are separate now
- https://observablehq.com/@d3/d3-group
Scivis and Infovis

- Two subfields of visualization
  - **Scivis** deals with data where the spatial position is given with data
    - Usually continuous data
    - Often displaying physical phenonema
    - Techniques like isosurfacing, volume rendering, vector field vis
  - **Infovis**, the data has no set spatial representation, designer chooses how to visually represent data
SciVis

[Google Image Search for "scientific visualization", 2017]
InfoVis

[Google Image Search for "information visualization", 2017]
Fields

- Values come from a **continuous** domain, infinitely many values
- **Sampled** at certain positions to approximate the entire domain
- Positions are often aligned in **grids**
- Often measurements of natural or simulated phenomena
- Examples: temperature, wind speed, tissue density, pressure, speed, electrical conductance
Fields in Visualization

Scalar Fields
(Order-0 Tensor Fields)

Vector Fields
(Order-1 Tensor Fields)

Tensor Fields
(Order-2+)

Each point in space has an associated...

Scalar

\[ s_0 \]

Vector

\[
\begin{bmatrix}
  v_0 \\
  v_1 \\
  v_2 
\end{bmatrix}
\]

Tensor

\[
\begin{bmatrix}
  \sigma_{00} & \sigma_{01} & \sigma_{02} \\
  \sigma_{10} & \sigma_{11} & \sigma_{12} \\
  \sigma_{20} & \sigma_{21} & \sigma_{22}
\end{bmatrix}
\]
Grids

- Remember we have continuous data and want to sample it in order to understand the entire domain
- Possible schemes?

- Geometry: the spatial positions of the data (points)
Grids

- Remember we have continuous data and want to sample it in order to understand the entire domain
- Possible schemes?
  - Geometry: the spatial positions of the data (points)
  - Topology: how the points are connected (cells)
  - Type of grid determines how much data needs to be stored for both geometry and topology

Grid types

- uniform
- rectilinear
- structured
- unstructured

© Weiskopf/Machiraju/Möller
Visualizing Volume (3D) Data

- 2D visualization
  slice images
  (or multi-planar reformating MPR)

- Indirect
  3D visualization
  isosurfaces
  (or surface-shaded display SSD)

- Direct
  3D visualization
  (direct volume rendering DVR)

[© Weiskopf/Machiraju/Möller]
Data

- In this lecture, we will be considering **scalar** data: a single value at each point.
- Our data is always discrete, what is the value of a point not exactly on our grid?
- Need a method to determine what these values are: interpolation schemes.
Interpolation

Value at 2.2?
Nearest Neighbor Interpolation

Value at 2.2?
Linear Interpolation

Value at 2.2?
Interpolation

• Other schemes:
  - polynomial interpolation
  - splines
  - more…
Dimensions of Data

• 1-Dimension: data along a line
  - Example: temperature along my drive from Tucson to Dartmouth
• 2-Dimensional: data on a plane
  - Example: temperature on the surface of a pond
• 3-Dimensional: data in our normal world (data in a volume)
  - Example: temperature at every point in the room
• Complexity increases as we add dimensions
• Visualization complexity also increases
• Often, want to be able to see phenomena as we see them in real life settings
3D: Voxels and Cells

- **Voxel**: grid point in center, constant value in voxel
- **Cell**: grid points at vertices, value within cell varies

Visualizing Volume (3D) Data

- **2D visualization slice images** (or multi-planar reformating MPR)
- **Indirect**
  - 3D visualization isosurfaces (or surface-shaded display SSD)
- **Direct**
  - 3D visualization (direct volume rendering DVR)

[© Weiskopf/Machiraju/Möller]
Visualizing Volume (3D) Data

(a) 2D slice

(b) Volume Rendering

[J. Kniss, 2002]
Visualizing Volume (3D) Data

(a) 2D slice

(b) Volume Rendering

[J. Kniss, 2002]
Visualizing Volume (3D) Data

(a) An isosurfaced tooth.

(b) Multiple isosurfaces. [J. Kniss, 2002]
How have we encoded 3D data before?  
Hint: Think about maps
Isolines (2D)

- Isoline: a line that has the same scalar value at all locations
- Example: Topographical Map
Isosurfaces (3D)

- Isosurface: a surface that has the same scalar value at all locations
- Often use multiple isosurfaces to show different levels

[J. Kniss, 2002]