Dynamic Visualizations (I)

Riccardo Pucella

October 19, 2015

Up until now

Static views

- unchanging data
- can have multiple static views on a page
- translates to poster visualizations

Next up: dynamic views

- update view based on *something* (programmed changes, user interaction)

Most useful when there is a lot of data

Dynamic views

Taxonomy of interactions for dynamic views:

- switching
- filtering
- refining (drill-down)
- ...

Rarely so clear cut:

- distinctions are often data-centric
- multiple interactions may co-exist

Switching

- Show different aspects of the data
  - those aspects are not hierarchical
- Essentially choosing one of several views
  - switching conceptually controlled by a selector
- We've seen these before!
  - example from class:
    charting 2013 versus 2014 Social Media data
  - exercise: modify project 2 to use switching
- Can construct views, or hide/show them

Demo

Filtering

- Useful when showing aggregate info about a lot of data

- Default: take all the data into account

- Filtering action: reduce the amount of data shown in the view
  - filter on some aspect of the data
  - predefined filters, user filters, etc.

Dealing with more complex data

- Social Media example:
  - data was pre-aggregated
  - dataset was small

- What does the original data look like?
  - survey data: one row per respondent
  - each row contains respondent info + answers
  - (similarly: event logs, ...)

- Process to update view:
  - filter data
  - aggregate (e.g., sum, percentages)
  - modify view based on new values

Pulling from CSV files

- Datasets often distributed as CSV files
  - if not, easily transformed
  - alt: pulling from a server (needed if data too big)

- Pulling from a CSV file in D3:
  - `d3.csv(`*`url`*`,`*`callback`*`)`
  - some trickiness when pulling from a local csv file
  - this is an asynchronous call: returns immediately
  - data is an array of objects

- The asynchrony may force you to think differently

Demo

Code structure of example

```
function run ()
function initializeView ()
function getDataRows (f)
function populateSelectors (data)
function setupEventListeners ()
function updateData ()
```

Code structu

```
function run ()
function initializ
function getDataRows
function populateSele
function setupEventLi
function updateData (
```

- set up the view
- get data
- fill selectors with values
- tell browser to update the view when data changes
- update the view with data

Code structu

```
function run ()
function initializ
function getDataRows
function populateSele
function setupEventLi
function updateData (
```

```
initializeView();
// asynchronous
getDataRows(function(data) {
   populateSelectors(data);
   setupEventListeners();
   DATA.all = data;
   updateData()
});
```

Code structure

As in previous examples layout the chart with zero values

```
function run ()
function initializeView ()
function getDataRows (f)
function populateSelectors (data)
function setupEventListeners ()
function updateData ()
```

Code structure of example

```
function run ()
function initializeView ()
function getDataRows (f)
function populateSele
function setupEven
function updateData
```

```
d3.csv("data.csv",
      function(error,data) {
         console.log(data);
         f(data);
});
```

Note that `d3.csv` calls the function *f* provided as input with the retrieved data

Code structure of example

```
function run ()
function initializeView ()
function getDataRows (f)
function populateSelectors (data)
function setupEventListeners ()
function updateData ()
```

Fill the three selectors with the values retrieved from the data

Typical JavaScript manipulation of HTML elements

Code structure of example

```
function run ()
function initializeView ()
function getDataRows (f)
function populateSelectors (data)
function setupEventListeners ()
function updateD    ()
```

Attach event listeners to all three selectors

Event `change` will trigger `updateData()`

Code structure of example

```
function run ()
function initializeView ()
function getDataRows (f)
function populateSelectors (data)
function setupEventListeners ()
function updateData ()
```

- Filter the data based on content of the selectors
- Compute aggregate counts
- Updating view elements with the results

Next time

- refining (drill-down)
- fun with events