Static Visualizations (I)

Riccardo Pucella

September 28, 2015

A visualization spectrum

Explorative

Characteristics

- minimalist
- only includes elements that represent data
- seeks to communicate information in the most clear, concise manner

Applications

- academic research
- science
- business intelligence
- data analysis

Narrative

Characteristics

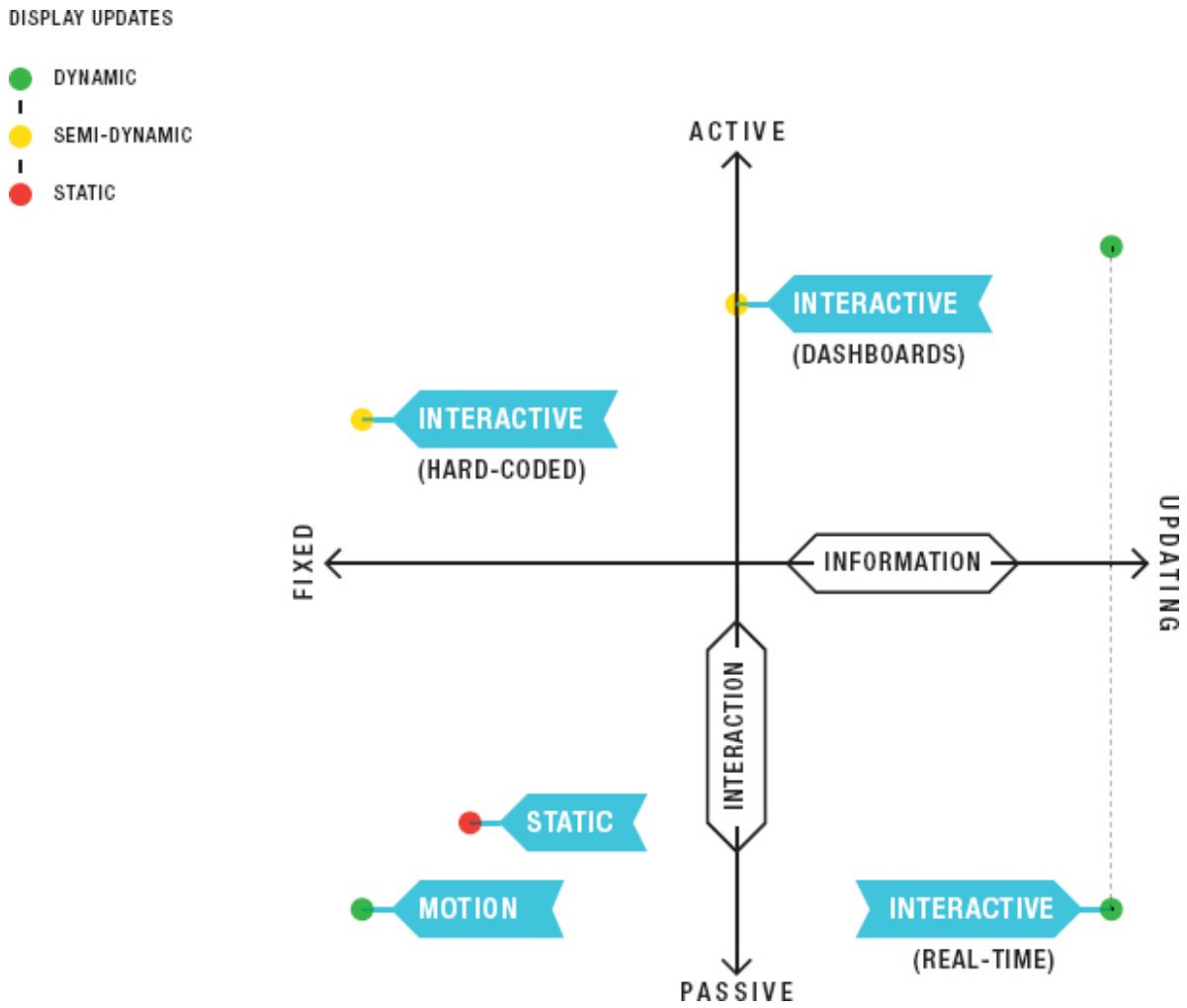
- illustrative
- design-focused
- seeks to appeal to viewer by engaging visuals
- informs and entertains

Applications

- publications
- blogs
- content marketing
- sales and marketing materials

Source: Lankov, Ritchie, Crooks, *Infographics* (2012)

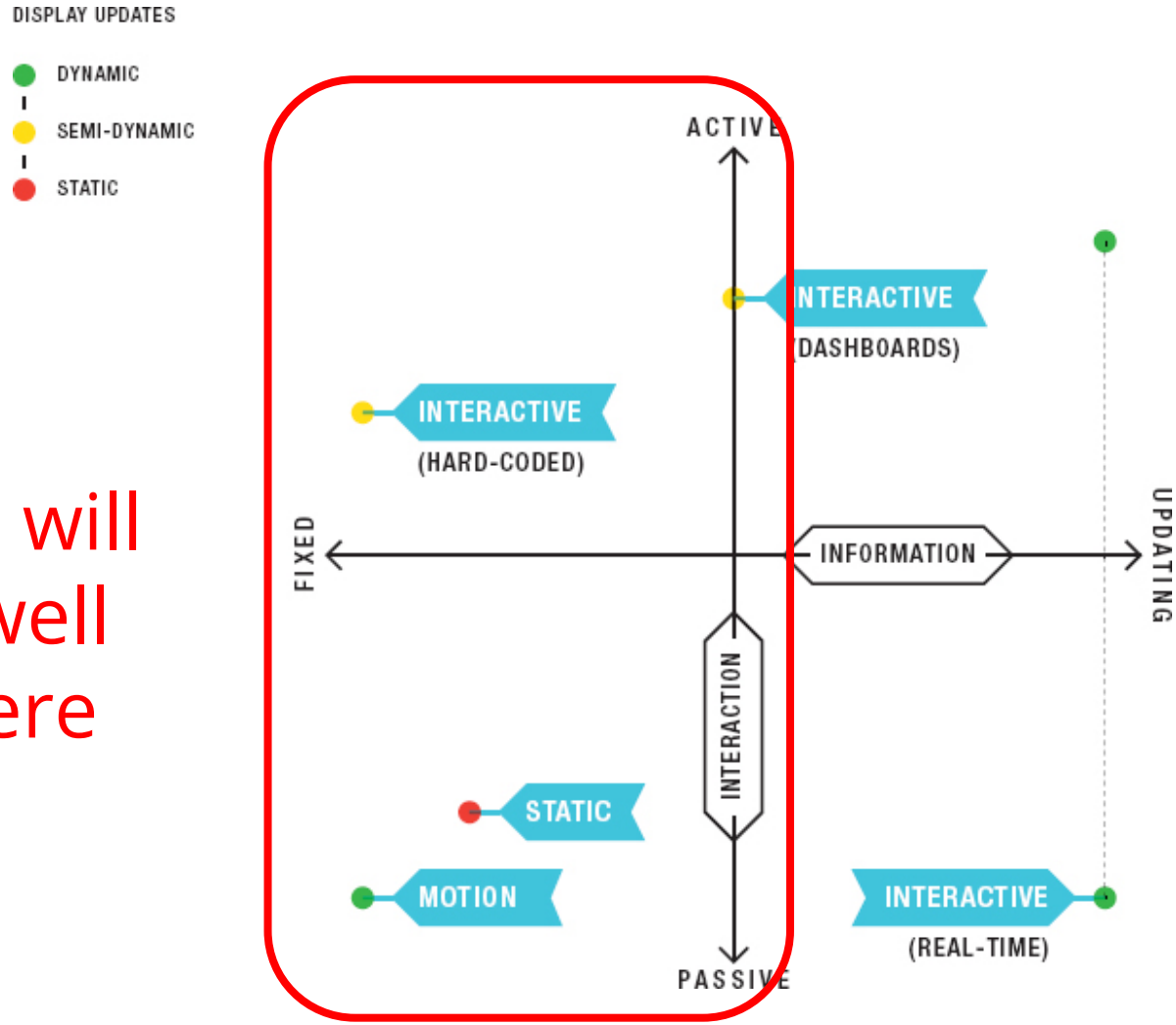
Infographics formats quadrant



Source: Lankov, Ritchie, Crooks, *Infographics* (2012)

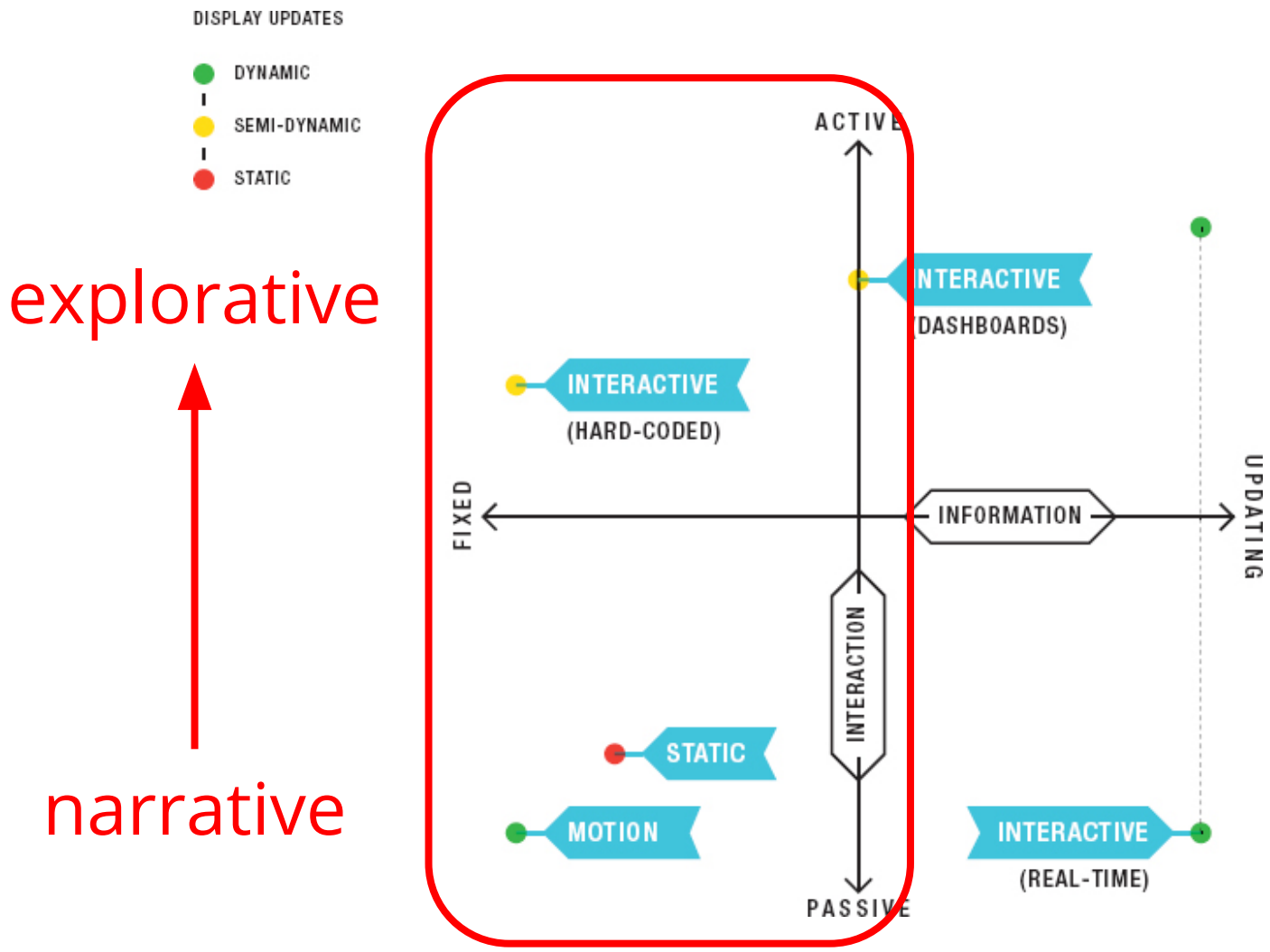
Infographics formats quadrant

We will dwell here



Source: Lankov, Ritchie, Crooks, *Infographics* (2012)

Infographics formats quadrant



Source: Lankov, Ritchie, Crooks, *Infographics* (2012)

Quantitative data

Graphs:
Representation of the relationships in *quantitative* information

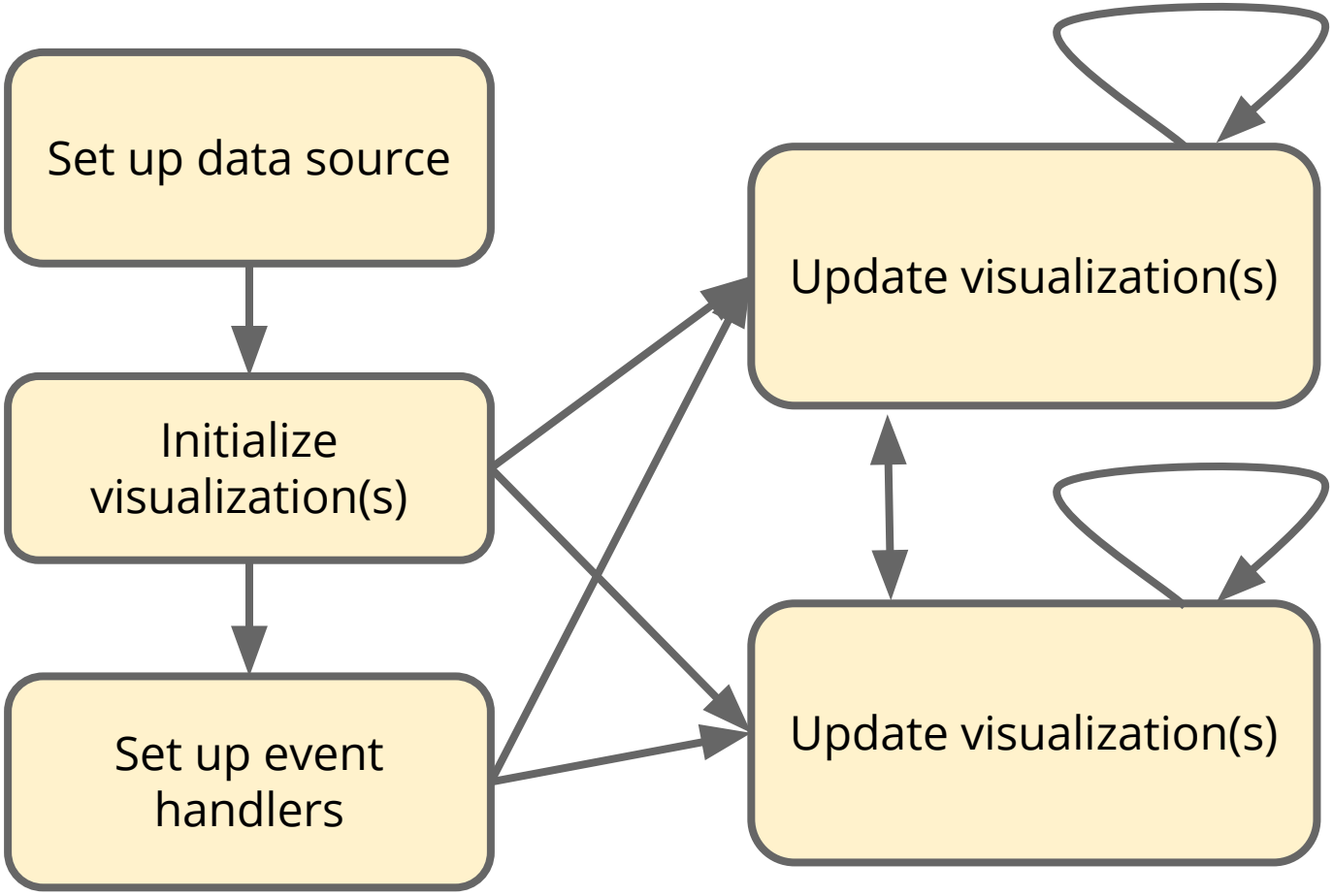
Most common types of relationships:

- nominal comparisons
- time series
- ranking
- part-to-whole

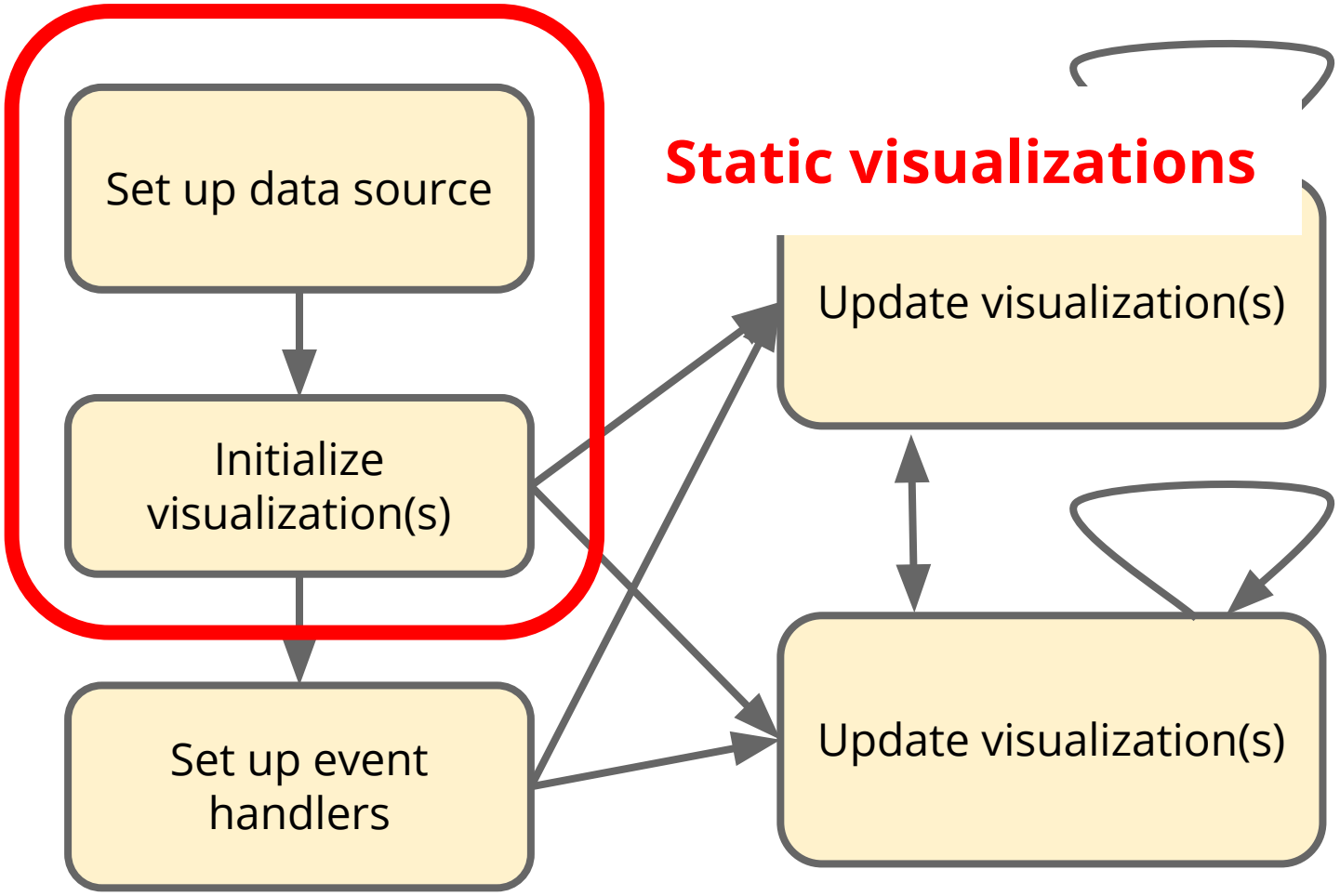
Quantitative data

- Dot charts / scatter plots
  - mostly nominal (dot charts), time series (discrete)
- Line charts
  - time series (continuous)
- Bar graphs (vertical, horizontal)
  - often nominal, ranking (horiz), time (vert)
- Stacked bar charts
  - multiple part-to-whole
- Pie charts
  - part-to-whole
- Bubble charts
  - nominal, ranking

Basic visualization architecture



Basic visualization architecture



Example: Social Media usage

Mobile Messaging and Social Media 2014
(Pew Research Center)

Simple charts are available in packages:

- Google Charts
- HighCharts
- etc...
- basic website dashboarding needs

(If that's all you care about, stop now)

Example:

Source: http://www.pewinternet.org/2015/01/09/social-media-update-2014/

Facebook users

Among online adults, the % who use Facebook

	2013	2014
All internet users	*71%*	*71%*
Men	66	66
Women	76	77
White, Non-Hispanic	71	71
Black, Non-Hispanic	76	67
Hispanic	73	73
18-29	84	87
30-49	79	73
50-64	60	63
65+	45	56*
High school grad or less	71	70
Some college	75	71
College+ (n=685)	68	74*
Less than $30,000/yr	76	77
$30,000-$49,999	76	69
$50,000-$74,999	68	74
$75,000+	69	72
Urban	75	71
Suburban	69	72
Rural	71	69

Source: Pew Research Center's Internet Project September Combined Omnibus Survey, September 11-14 & September 18-21, 2014. N=1,597 internet users ages 18+. The margin of error for all internet users is +/- 2.9 percentage points. 2013 data from Pew Internet August Tracking Survey, August 07 – September 16, 2013, n= 1,445 internet users ages 18+.

Note: Percentages marked with an asterisk (*) represent a significant change from 2013. Results are significant at the 95% confidence level using an independent z-test.

PEW RESEARCH CENTER

About data

Two questions:

- How do we structure it?
- How do we access it?

About data

Two questions:

- How do we structure it?
- How do we access it?

Data structuring a huge topic

- Excel-like multi-dimensional tables?
- SQL-like row-based tables?
- JSON-based structured data?

Sometimes need to go back and forth

About data

Two questions:

- How do we structure it?
- How do we access it?

How to access data really depends on size:

- do we load the data all at once?
- or do we pull only the slice of data needed from the data source upon request?

Abou

Two questions:

- How do we structur
- How do we access i

Abstract away from this by using a function (or an object) to access the data

How to access data really depends on size:

- do we load the data all at once?
- or do we pull only the slice of data needed from the data source upon request?

Example: the data

Let’s use a row-based representation:

```
[{ category:"all", group:"all", year:2013, value:71},
 { category:"all", group:"all", year:2014, value:71},
 { category:"gender", group:"Men", year:2013, value:66},
 { category:"gender", group:"Men", year:2014, value:66},
 { category:"gender", group:"Women", year:2013, value:76},
 { category:"gender", group:"Women", year:2014, value:77},
 { category:"race", group:"White, Non-Hispanic", year:2013, value:71},
 { category:"race", group:"White, Non-Hispanic", year:2014, value:71},
 { category:"race", group:"Black, Non-Hispanic", year:2013, value:76},
 { category:"race", group:"Black, Non-Hispanic", year:2014, value:67},
 { category:"race", group:"Hispanic", year:2013, value:73},
 { category:"race", group:"Hispanic", year:2014, value:73},
 { category:"age", group:"18-29", year:2013, value:84},
 … ]
```

Example: Choosing a chart

Let’s keep things simple.

The data is categorical

- split by category: age 18-29, 30-49, ...
- for each category, a single value
- Spread across two years

Histogram: emphasizes the categories

Alternatives?

Building a histogram by hand

For each data element:

- draw a bar of the right size
- show the data value
- label the bar

One year: `social-media-1.html`

Multiple years: `social-media-2.html`

D3

A JS library for manipulating the DOM following the structure of supplied data

- D3 = Data-Driven Documents

Derived from the ProtoVis project at Stanford

Not restricted to data visualization, but it has nice features for building visualizations

D3 for DOM manipulation

There's a cottage industry of JS libraries to help make DOM manipulation easier

- jQuery the common example
- `$("#some-element")` vs `document.getElementById("some-element")`

D3 plays that role as well

- and it's tied in nicely with data manipulation

D3 element manipulation

```
var svg = d3.select(“#viz”);

var height = svg.attr(“height);

var rect = svg.append(“rect”);
rect.attr(“x”,0);
rect.attr(“y”,0);
rect.attr(“width”,100);
rect.attr(“height”,100);
rect.style(“fill”,”blue”);
```

D3 element manipulation

```
var svg = d3.select("#viz");

var height = svg
var rect = s
rect.attr("x
rect.attr("y
rect.attr("w
rect.attr("height",100);
rect.style("fill","blue");
```

Get element with ID viz from the DOM

D3 element manipulation

```
var svg = d3.select(“#viz”);

var height = svg.attr(“height);

var rect = svg.ap                (“rect”);
rect.attr(“x”
rect.attr(“y
rect.attr(“w
rect.attr(“h
rect.style(“
```

Look up the attribute of a selected element

D3 element manipulation

```
var svg = d

var heigh
```

Create a new child element of selected element

```
var rect = svg.append("rect");
rect.attr("x",0);
rect.attr("y",0);
rect.attr("width",100);
rect.attr("height",100);
rect.style("fill","blue");
```

D3 element manipulation

```
var svg = d

var heigh

var rect = svg.append("rect");
rect.attr("x",0);
rect.attr("y",0);
rect.attr("width",100);
rect.attr("height",100);
rect.style("fill","blue");
```

Change attributes / style attributes of element

D3 element manipulation

```
var svg = d3.select(“#viz”);

var height = svg.attr(“height);

var rect = svg.append(“rect”)
              .attr(“x”,0)
              .attr(“y”,0)
              .attr(“width”,100)
              .attr(“height”,100)
              .style(“fill”,”blue”);
```

Method chaining!

D3 element manipulation

```
var svg = d3.select(“#viz”);

var height = svg.attr(“height);

var rect = svg.append(“rect”)
              .attr(“x”,0)
              .attr(“y”,0)
```

Rewritten in D3: social-media-3.html

Next time

- D3 selections and data binding
- Managing multiple static visualizations