DSCI 554 LECTURE 4

DESIGNING INFOGRAPHICS AND DASHBOARDS, D3 DATA JOIN BASICS AND LOADING DATA

Dr. Luciano Nocera

USC Viterbi
School of Engineering
Integrated Media Systems Center

OUTLINE

- Designing infographics and dashboards
- Function and esthetics, minimalistic visualizations
- D3 data join basics
- Loading data in D3

Diagrama

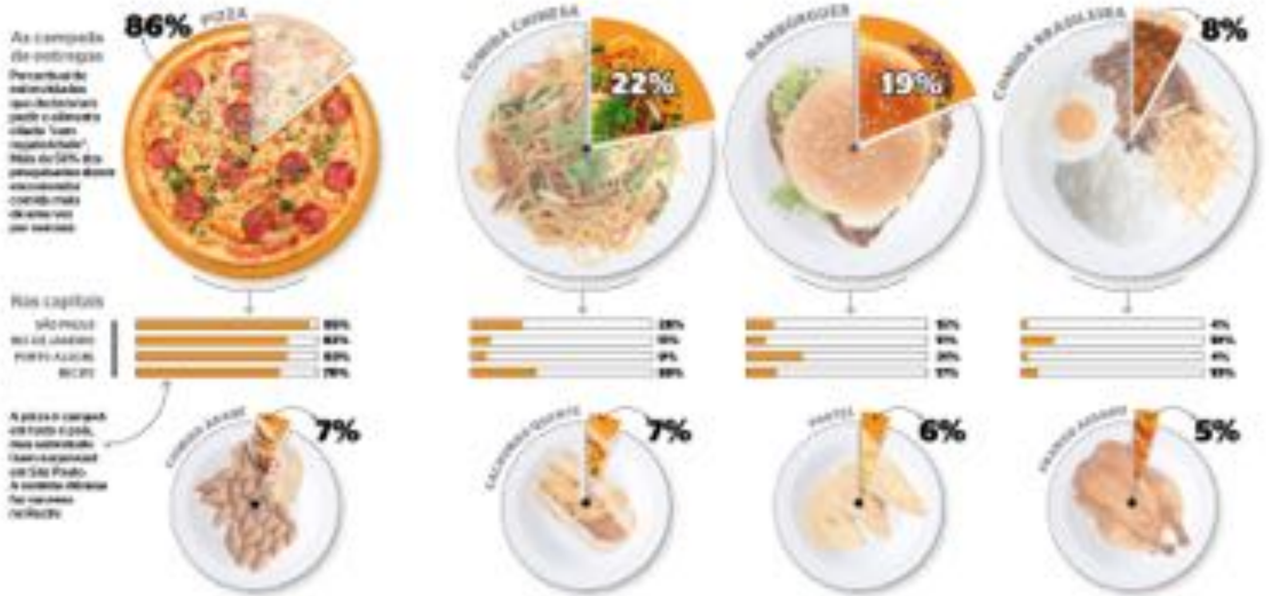
[illegible]

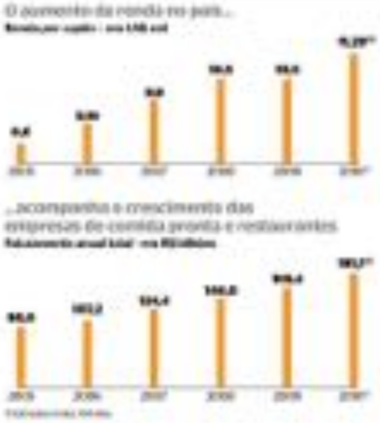
Quando o brasileiro come fora

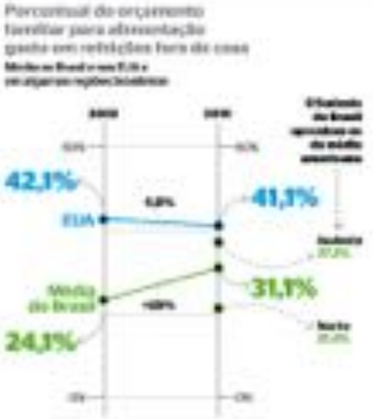
O crescimento da economia muda os hábitos alimentares e estimula o mercado de comida pronta

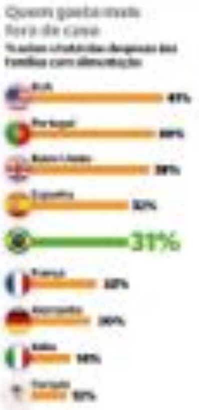
[illegible]

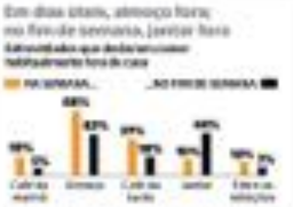
[illegible]

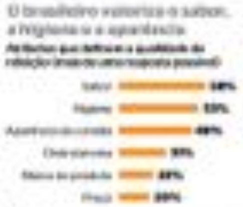


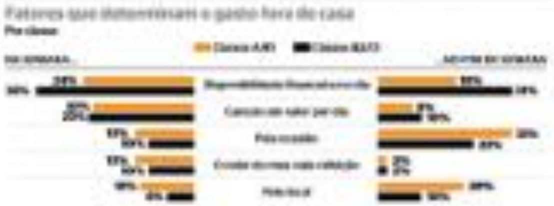






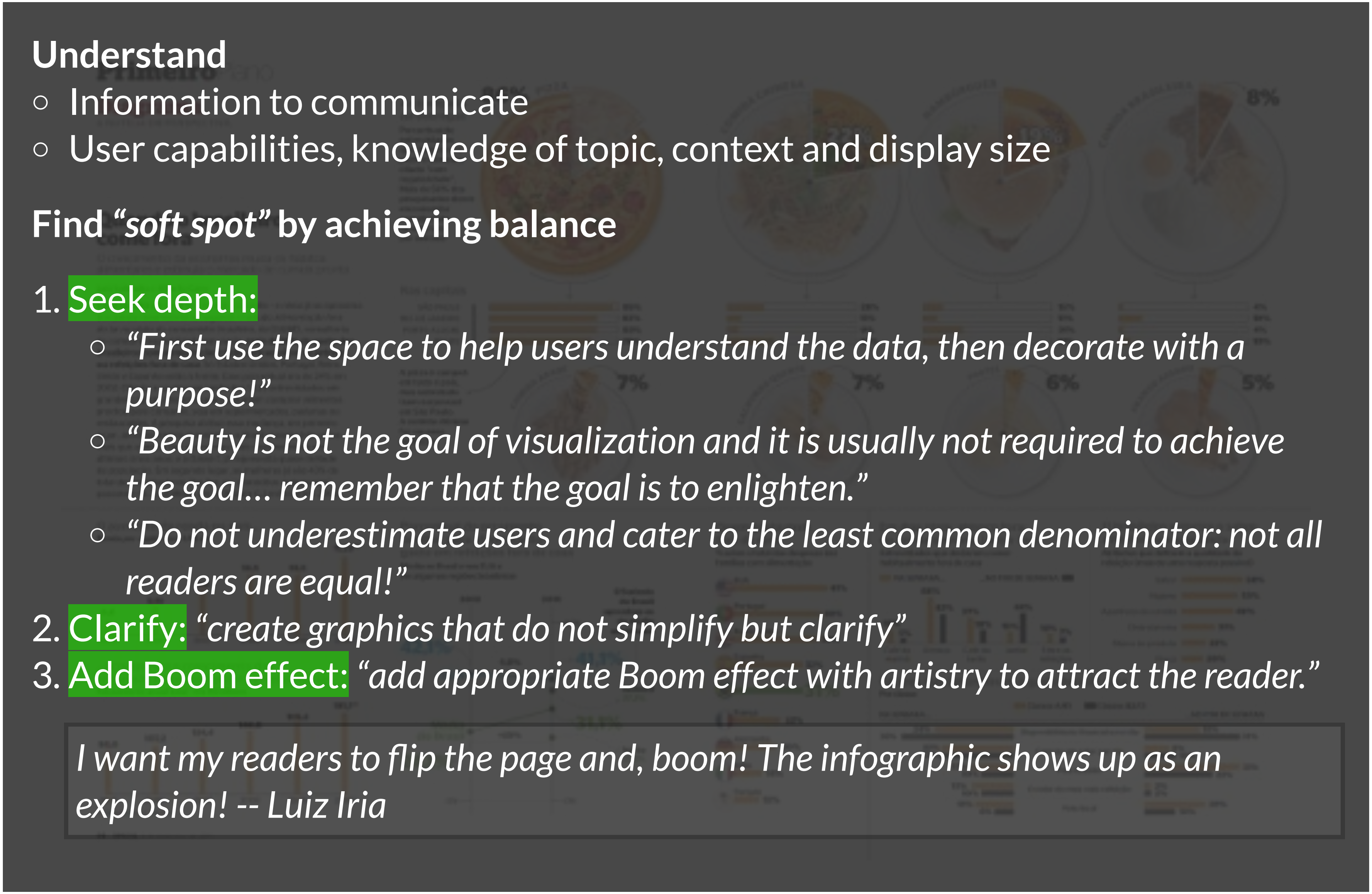






Understand

- Information to communicate
- User capabilities, knowledge of topic, context and display size



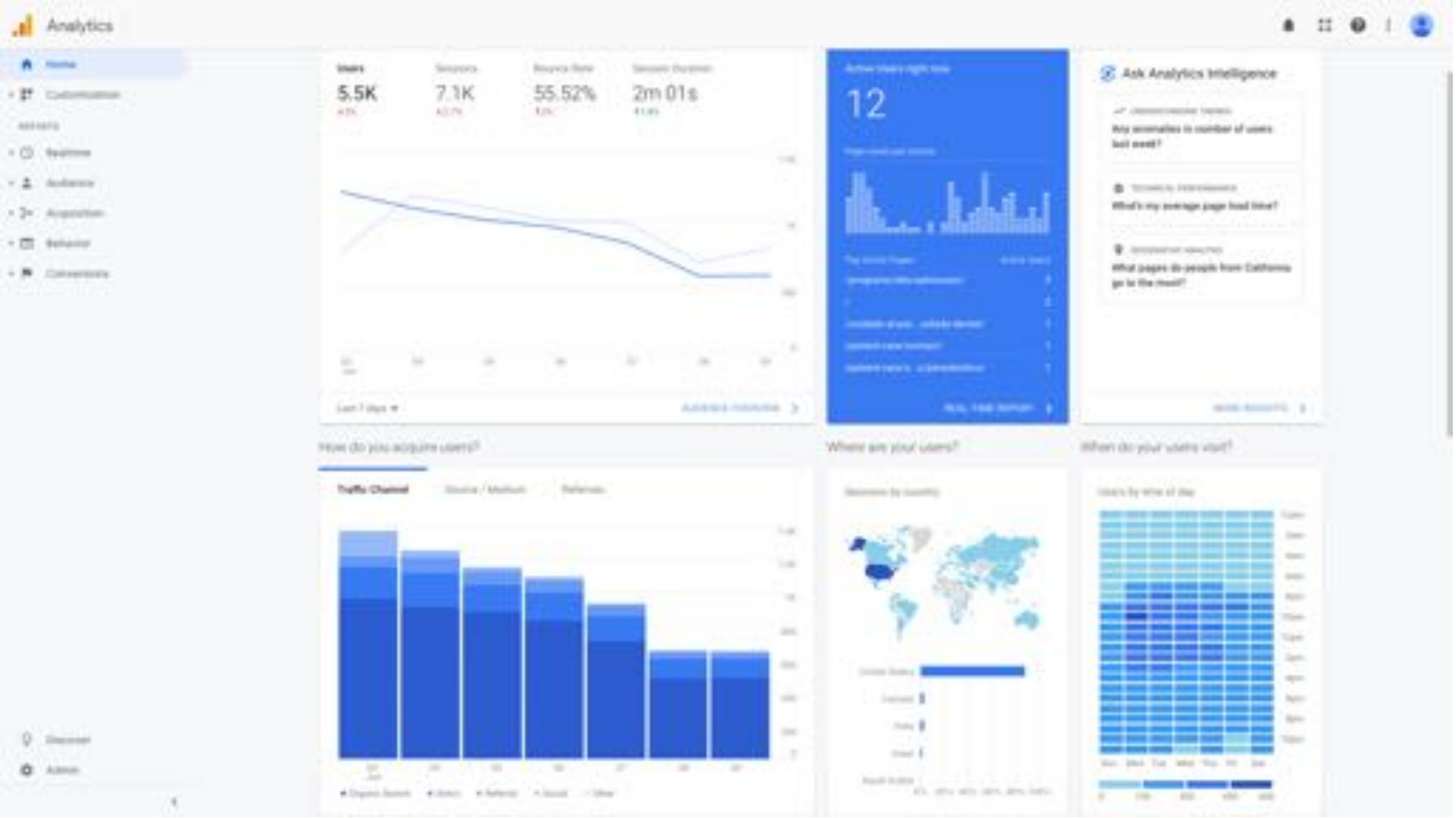
Understand

- Information to communicate
- User capabilities, knowledge of topic, context and display size

Find *"soft spot"* by achieving balance

1. Seek depth:
   - *"First use the space to help users understand the data, then decorate with a purpose!"*
   - *"Beauty is not the goal of visualization and it is usually not required to achieve the goal… remember that the goal is to enlighten."*
   - *"Do not underestimate users and cater to the least common denominator: not all readers are equal!"*
2. Clarify: *"create graphics that do not simplify but clarify"*
3. Add Boom effect: *"add appropriate Boom effect with artistry to attract the reader."*

> *I want my readers to flip the page and, boom! The infographic shows up as an explosion! -- Luiz Iria*



Users 5.5K

Sessions 7.1K

Bounce Rate 55.52%

Session Duration 2m 01s

Last 7 days

Active Users right now

12

Ask Analytics Intelligence

Any anomalies in number of users last week?

What's my average page load time?

What pages do people from California go to the most?

How do you acquire users?

Traffic Channel

Source / Medium

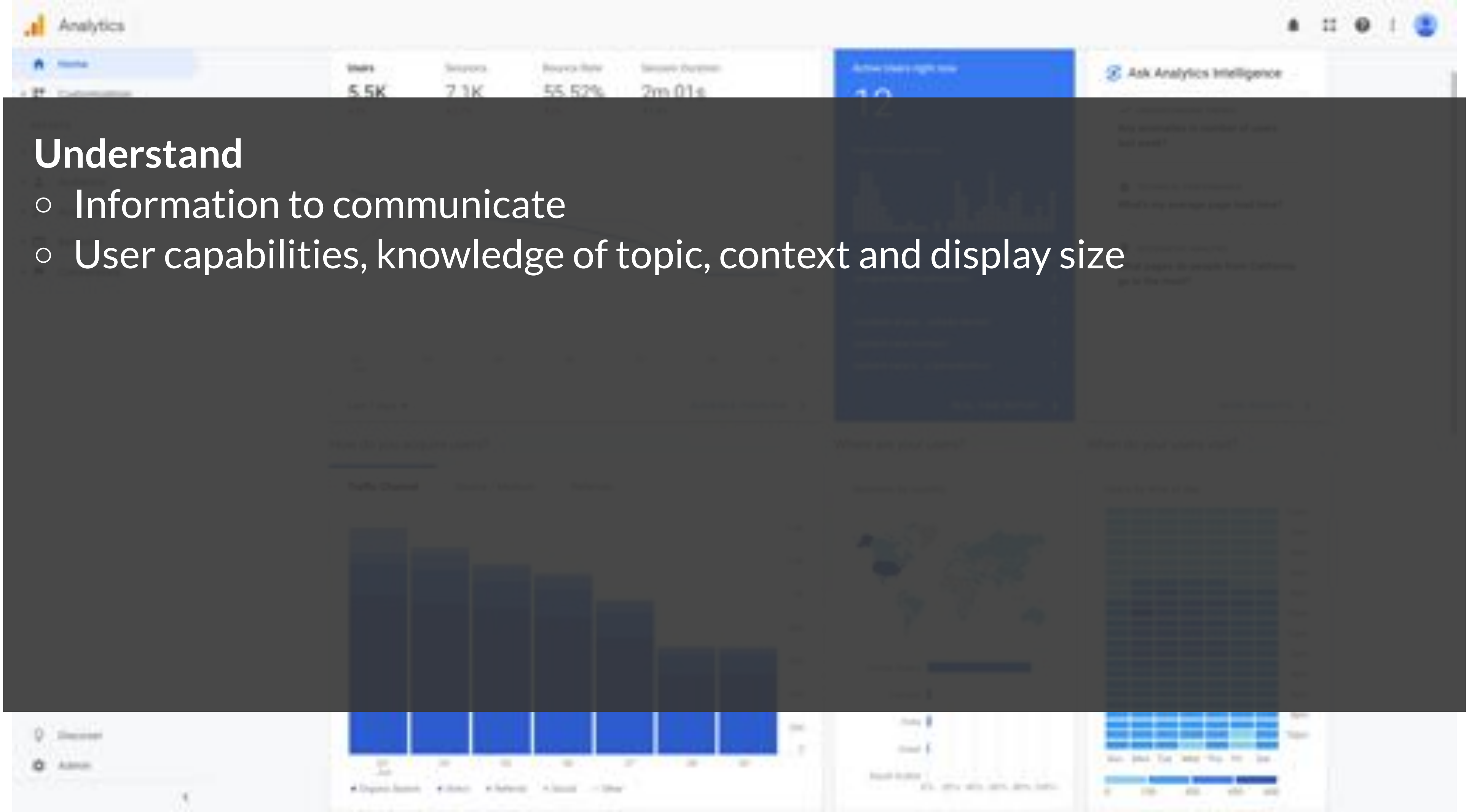
Referrals

Where are your users?

Sessions by country

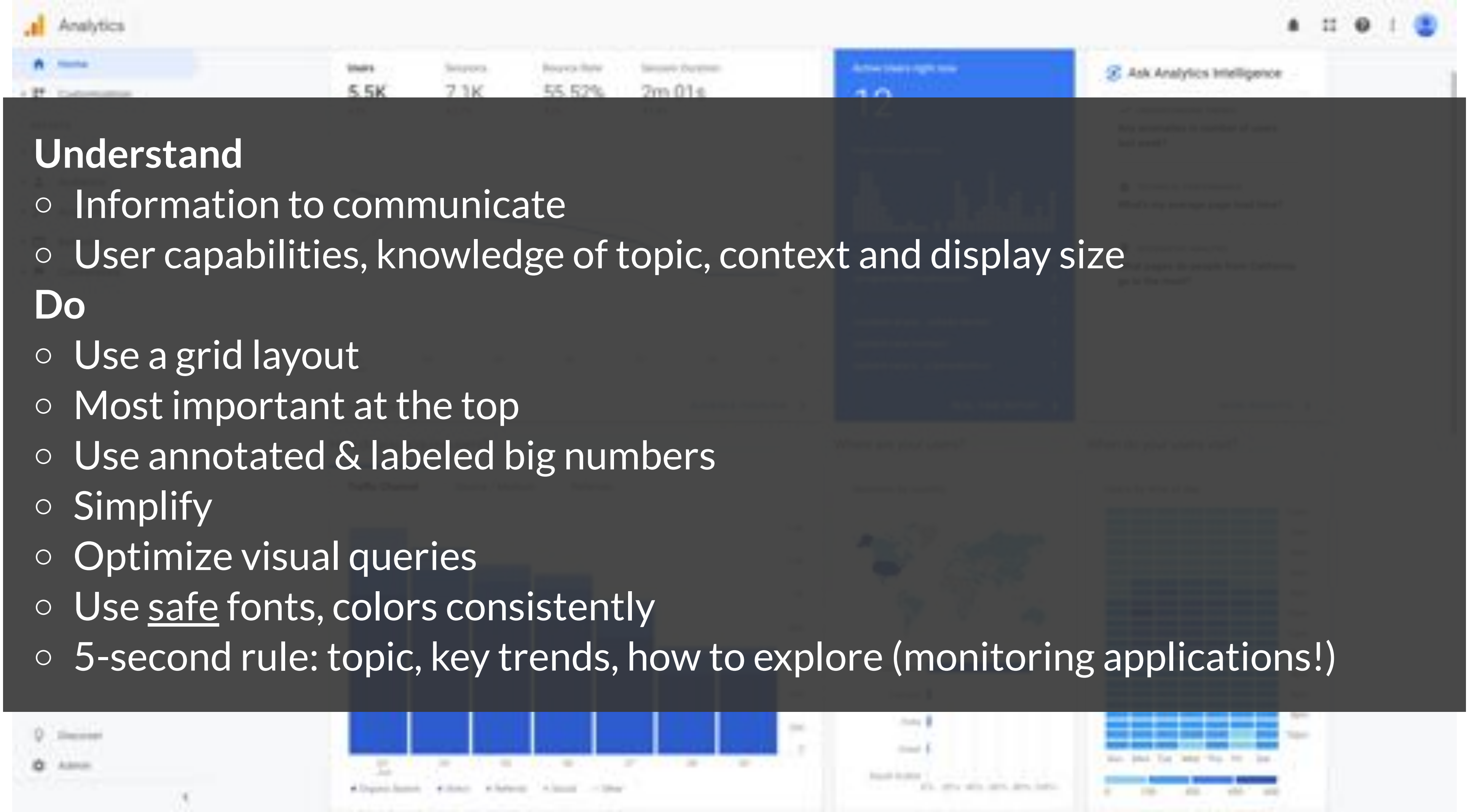
When do your users visit?

Users by time of day



Understand

- ○ Information to communicate
- ○ User capabilities, knowledge of topic, context and display size



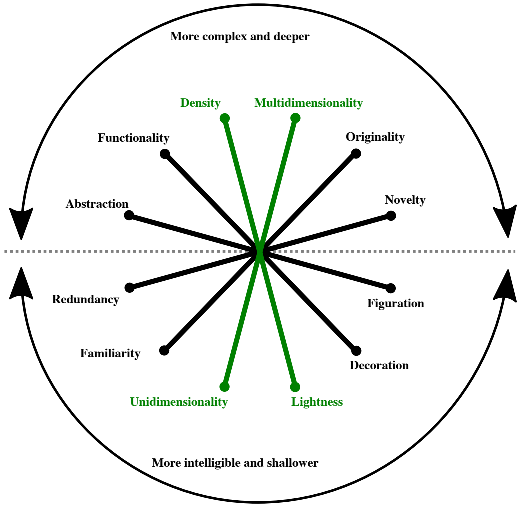
Understand

- Information to communicate
- User capabilities, knowledge of topic, context and display size

Do

- Use a grid layout
- Most important at the top
- Use annotated & labeled big numbers
- Simplify
- Optimize visual queries
- Use safe fonts, colors consistently
- 5-second rule: topic, key trends, how to explore (monitoring applications!)

1. SEEK DEPTH



CAIRO'S RECOMMENDATIONS

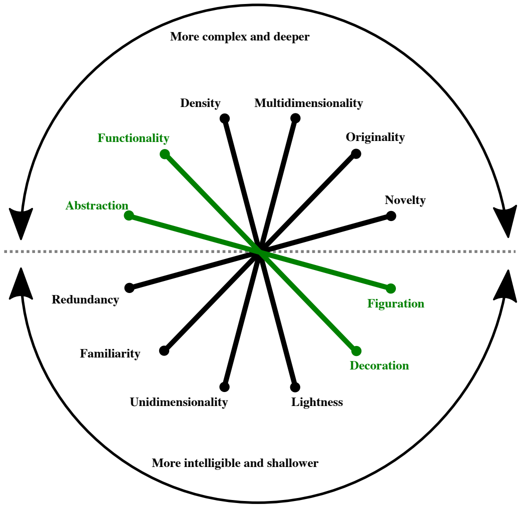
1. Define where your graphic stands in terms of density and dimensionality
2. Move position of graphic at least 10% towards density and multidimensionality
3. Organize in layers, starting with a summary
4. Include inner layers as necessary based on story and focus
5. Structure the layers in logical order

DASHBOARD RECOMMENDATIONS

- Use a grid layout
- Most important at the top
- Use annotated & labeled big numbers
- 5-second rule: topic, key trends, how to explore (monitoring applications!)

🟩 infographic & dashboard 🟪 infographic only 🟧 dashboard only

2. CLARIFY



CAIRO'S RECOMMENDATIONS

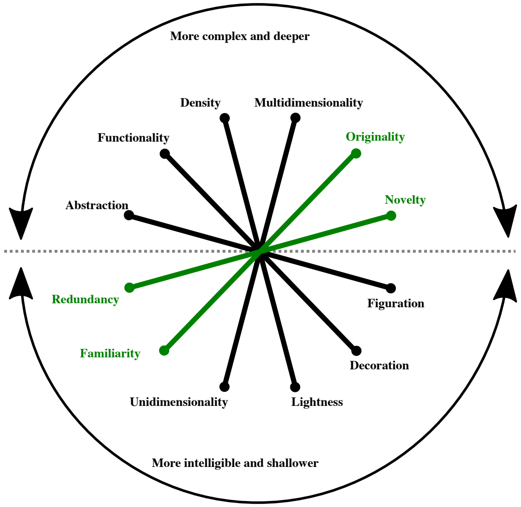
- Do not simplify but clarify
- Think about structure first then eye-candy
- Use space first to explain and develop the **story**
- Think about how data should be organized before thinking about style
- Never dumb down your data

DASHBOARD RECOMMENDATIONS

- Simplify
- Optimize visual queries
- 5-second rule: topic, key trends, how to explore (monitoring applications!)

🟩 infographic & dashboard 🟪 infographic only 🟧 dashboard only

3. BOOM EFFECT



CAIRO'S RECOMMENDATIONS

- Experiment (carefully) with novel (original) forms
- The more original the form the more redundancy
- Explain novel forms with text and other graphics

DASHBOARD RECOMMENDATIONS

- Use safe fonts and colors consistently
- 5-second rule: topic, key trends, how to explore (monitoring applications!)

🟩 infographic & dashboard 🟪 infographic only 🟧 dashboard only

OUTLINE

- Designing infographics and dashboards
- Function and esthetics, minimalistic visualizations
- D3 data join basics
- Loading data in D3

TUFTE'S DESIGN PRINCIPLE

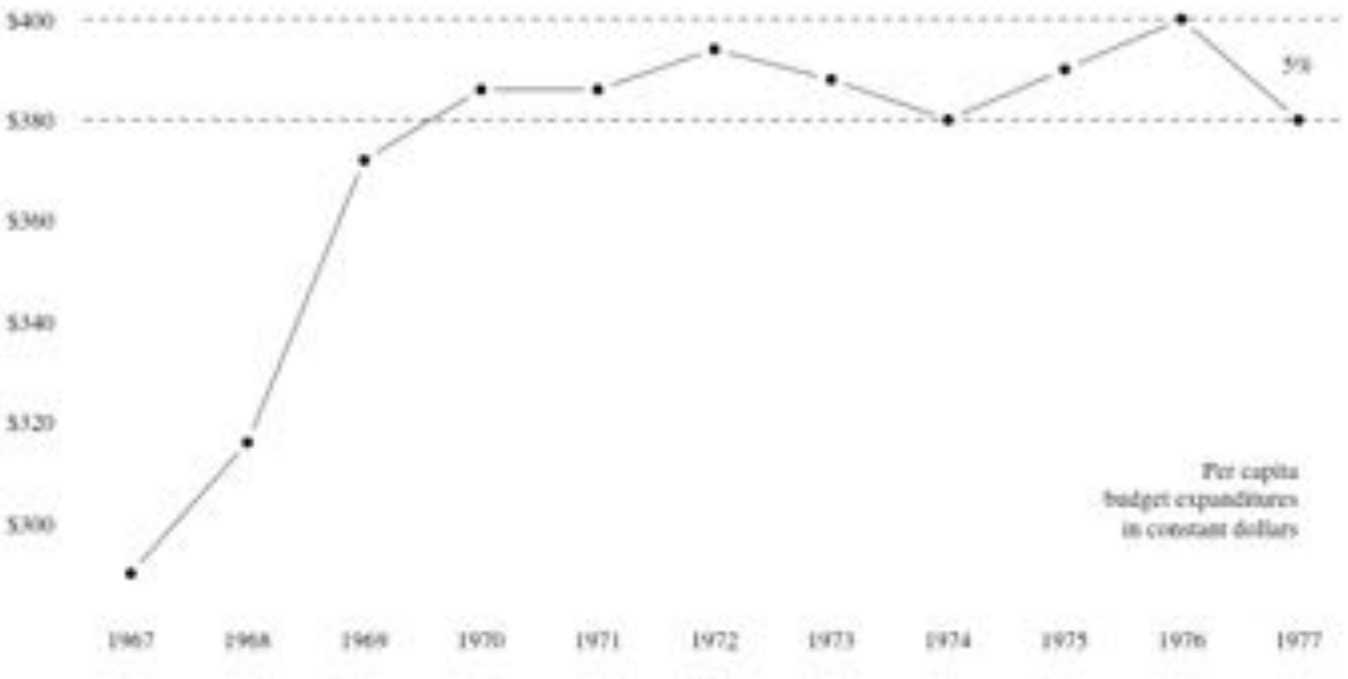
Elegance in visuals is attained when the complexity of the data matches the simplicity of the design

DATA-INK RATIO

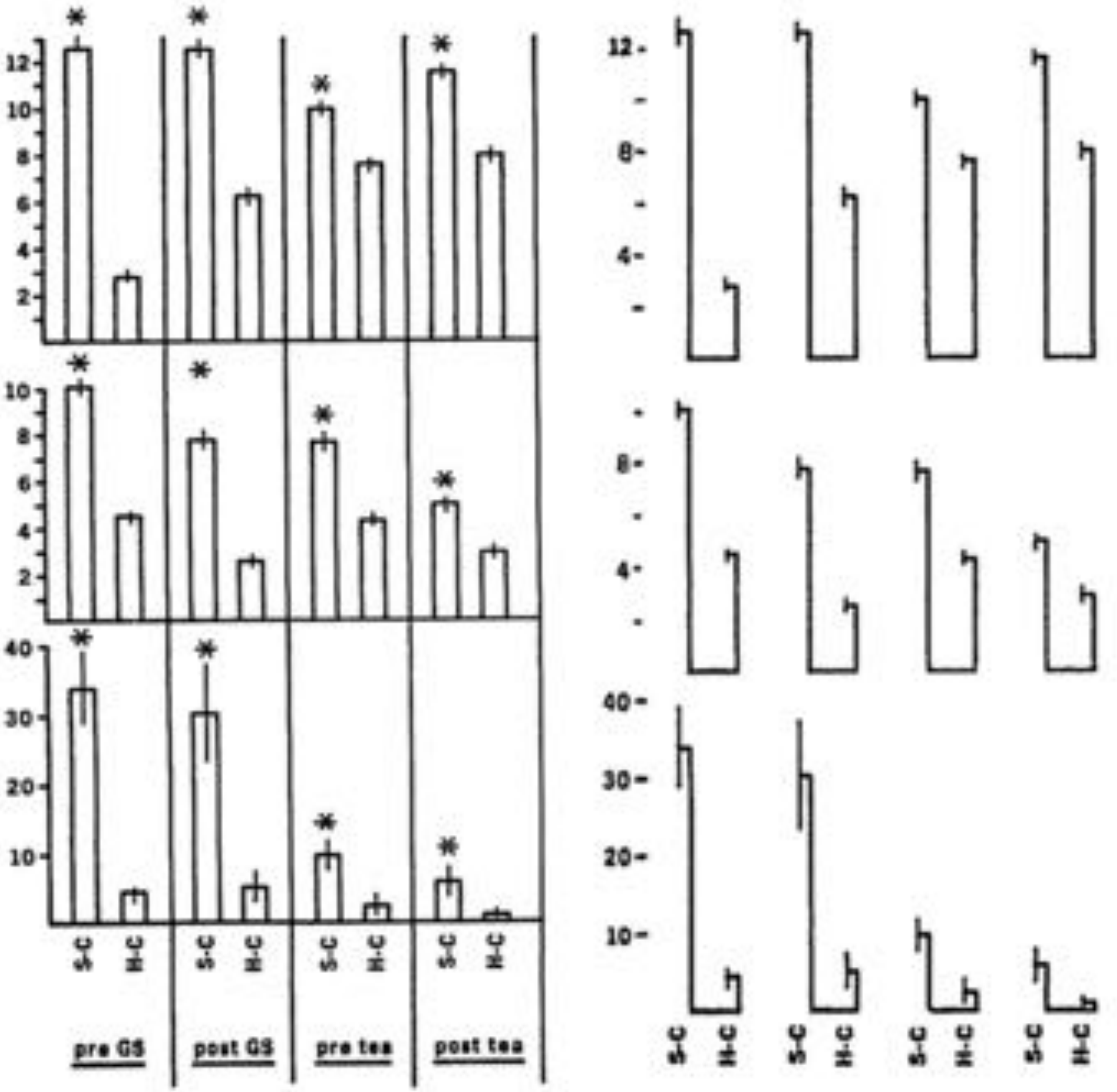
$$\begin{aligned}\text{Data-ink ratio} &= \frac{\text{Data-ink}}{\text{Total ink used to print the graphic}}\\ &= \text{Proportion of a graphic's ink devoted to the non-redundant display of data-information}\\ &= 1.0 - \text{Proportion of a graphic that can be erased without loss of data-information}\end{aligned}$$

TUFTE'S DESIGN PRINCIPLES

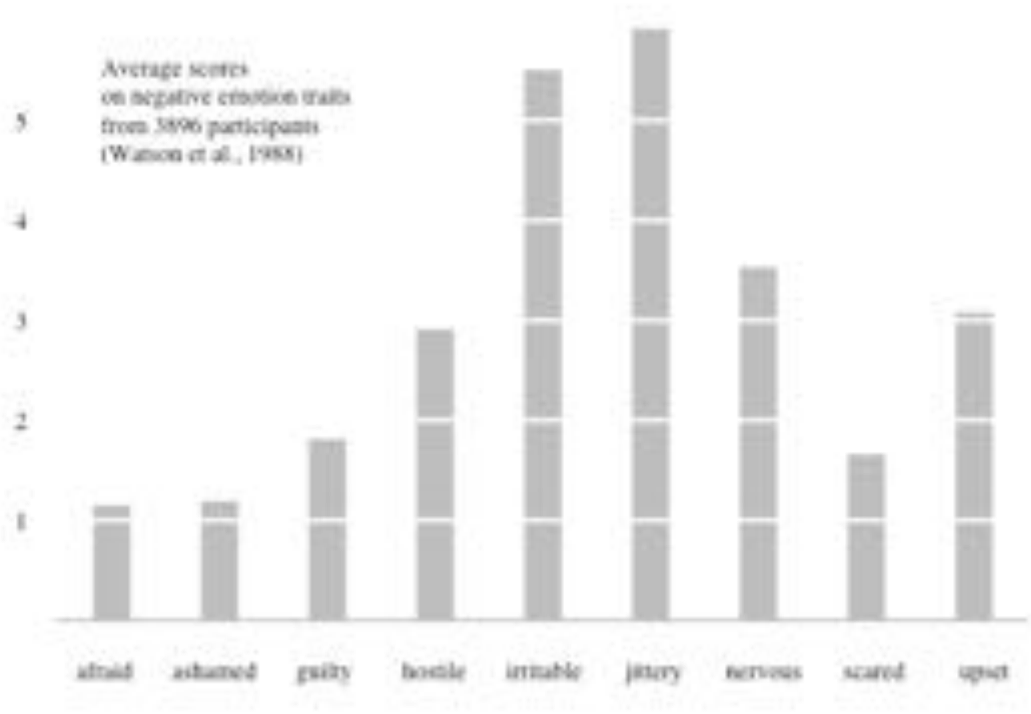
1. Above all else show data
2. Maximize the data-ink ratio
3. Erase non-data-ink
4. Erase redundant data-ink
5. Revise and edit



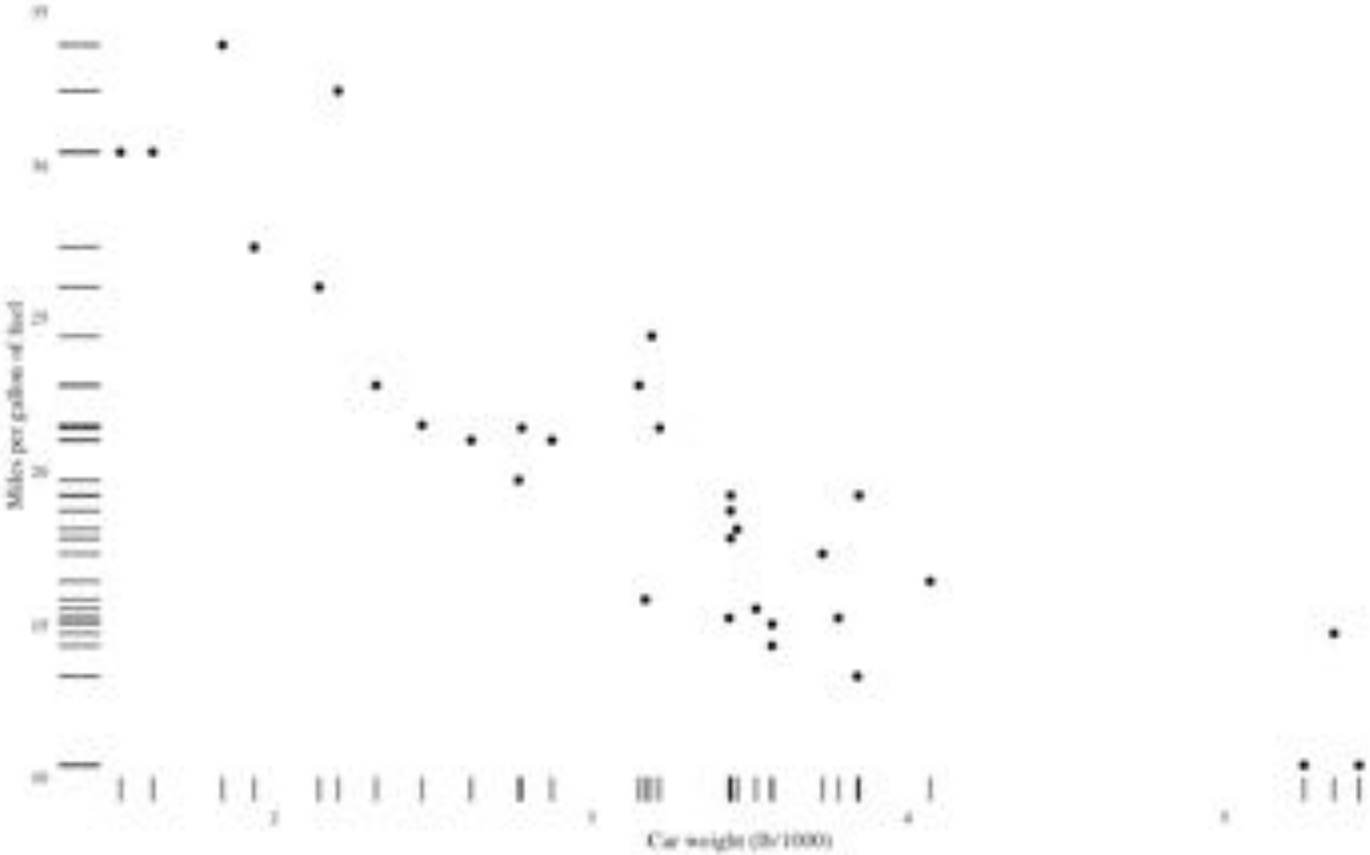
The Visual Display of Quantitative Information, E. Tufte, page 68



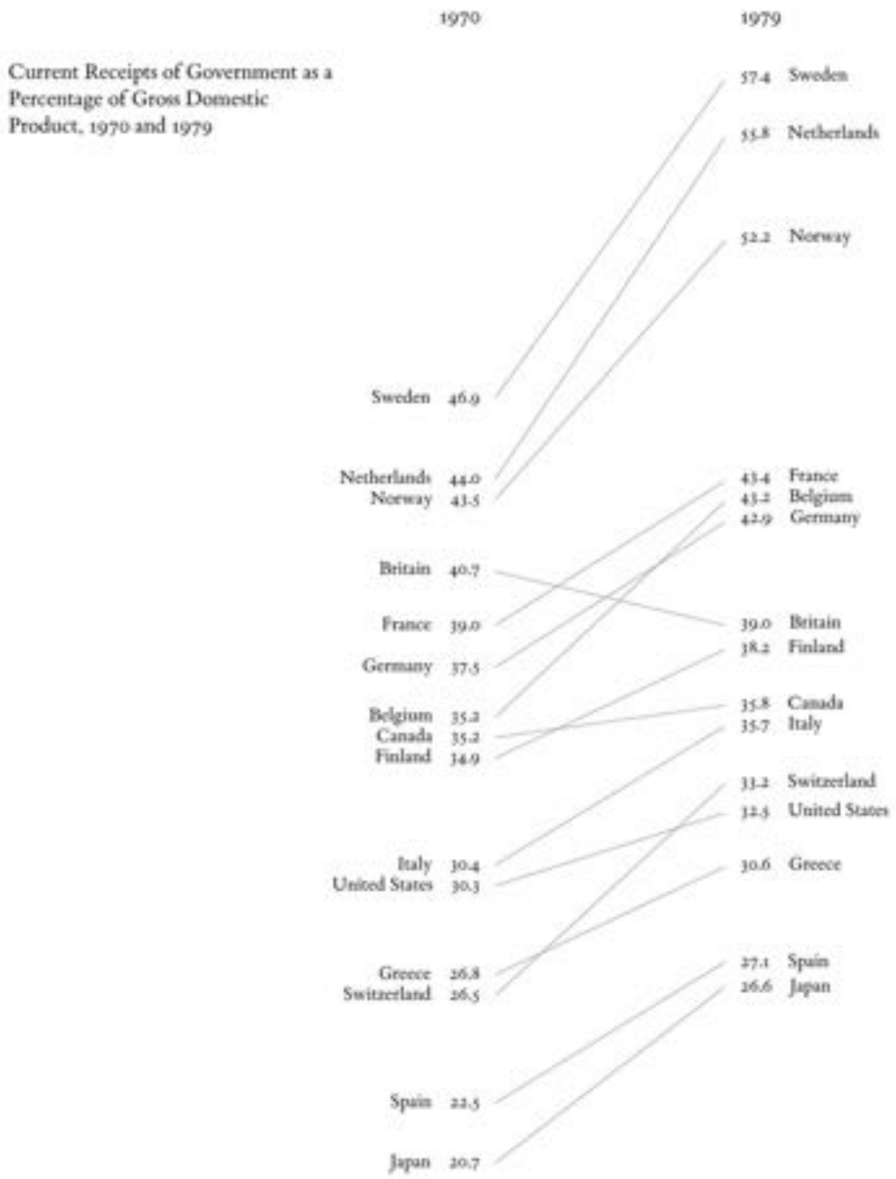
The Visual Display of Quantitative Information, E. Tufte, page 102



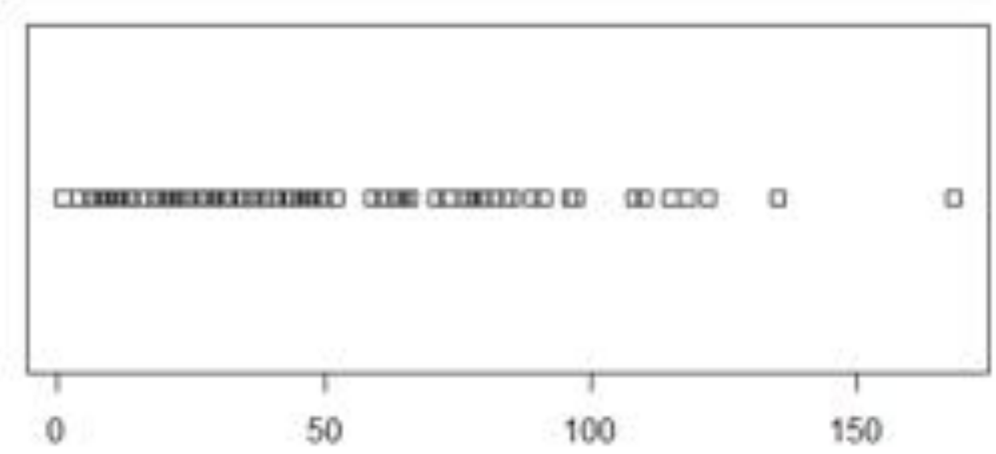
The Visual Display of Quantitative Information" p. 128



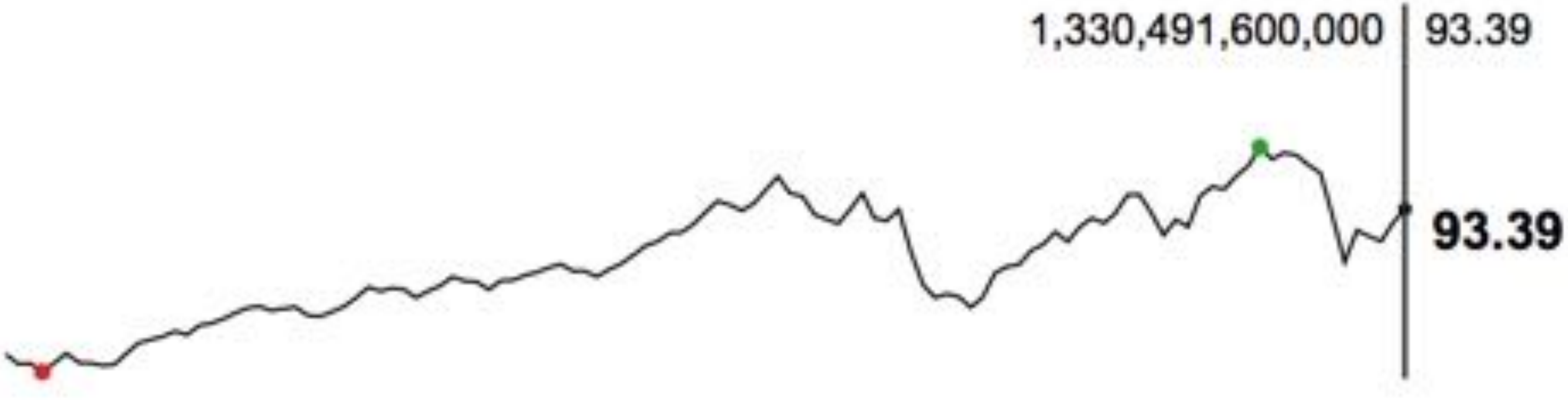
Dot-dash plot: The Visual Display of Quantitative Information, E. Tufte, page 133



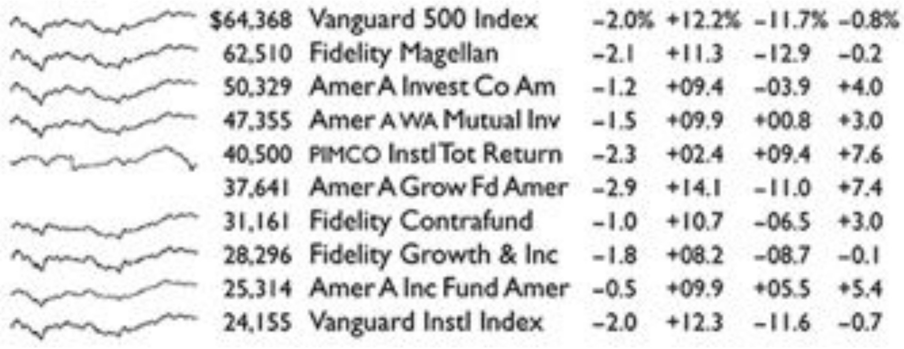
Slopegraph: The Visual Display of Quantitative Information" p. 158



Stripchart: 1-D scatter plot. Good alternative to boxplots when sample sizes are small.



Sparkline: line chart usually drawn without axes where the data is discussed



$64,368	Vanguard 500 Index	-2.0%	+12.2%	-11.7%	-0.8%
62,510	Fidelity Magellan	-2.1	+11.3	-12.9	-0.2
50,329	Amer A Invest Co Am	-1.2	+09.4	-03.9	+4.0
47,355	Amer A WA Mutual Inv	-1.5	+09.9	+00.8	+3.0
40,500	PIMCO Instl Tot Return	-2.3	+02.4	+09.4	+7.6
37,641	Amer A Grow Fd Amer	-2.9	+14.1	-11.0	+7.4
31,161	Fidelity Contrafund	-1.0	+10.7	-06.5	+3.0
28,296	Fidelity Growth & Inc	-1.8	+08.2	-08.7	-0.1
25,314	Amer A Inc Fund Amer	-0.5	+09.9	+05.5	+5.4
24,155	Vanguard Instl Index	-2.0	+12.3	-11.6	-0.7

Sparklines as small multiples

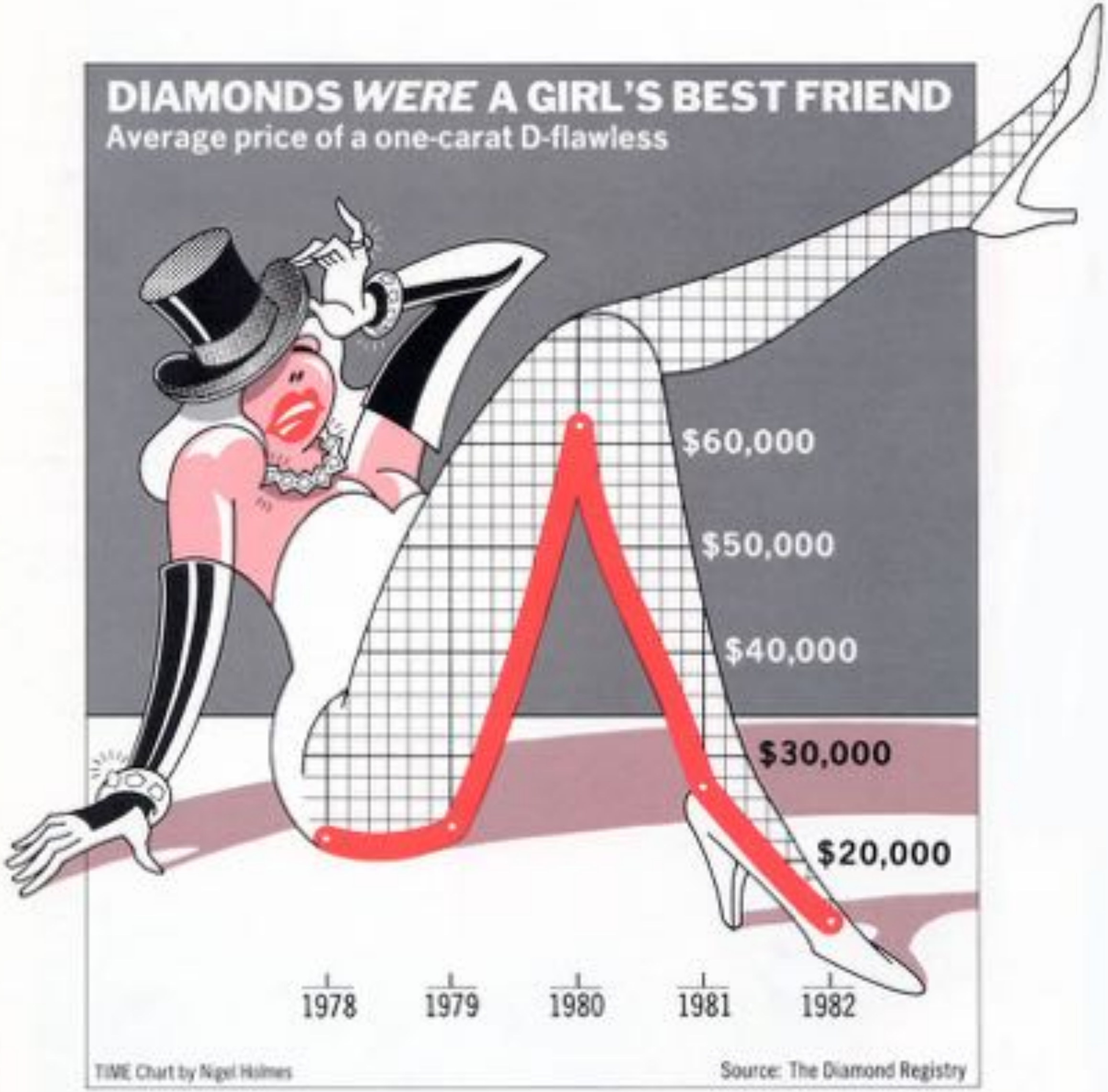
“CHARTJUNK”

The interior decoration of graphics generates a lot of ink that does not tell the viewer anything new.

The purpose of decoration varies — to make the graphic appear more scientific and precise, to enliven the display, to give the designer an opportunity to exercise artistic skills.

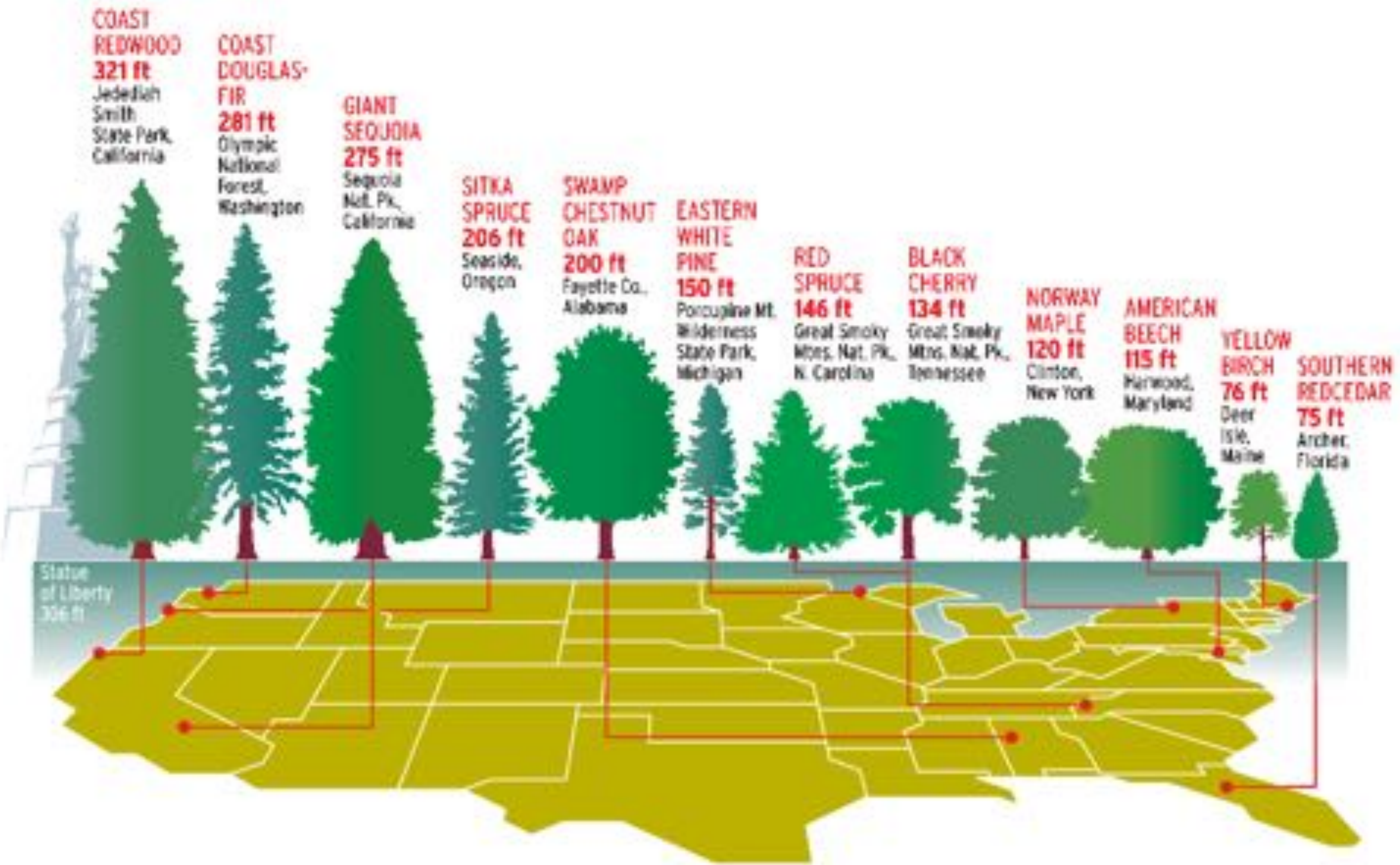
Regardless of its cause, it is all non-data-ink or redundant data-ink, and it is often chartjunk.

Tufte, Edward R. (1983). The Visual Display of Quantitative Information.

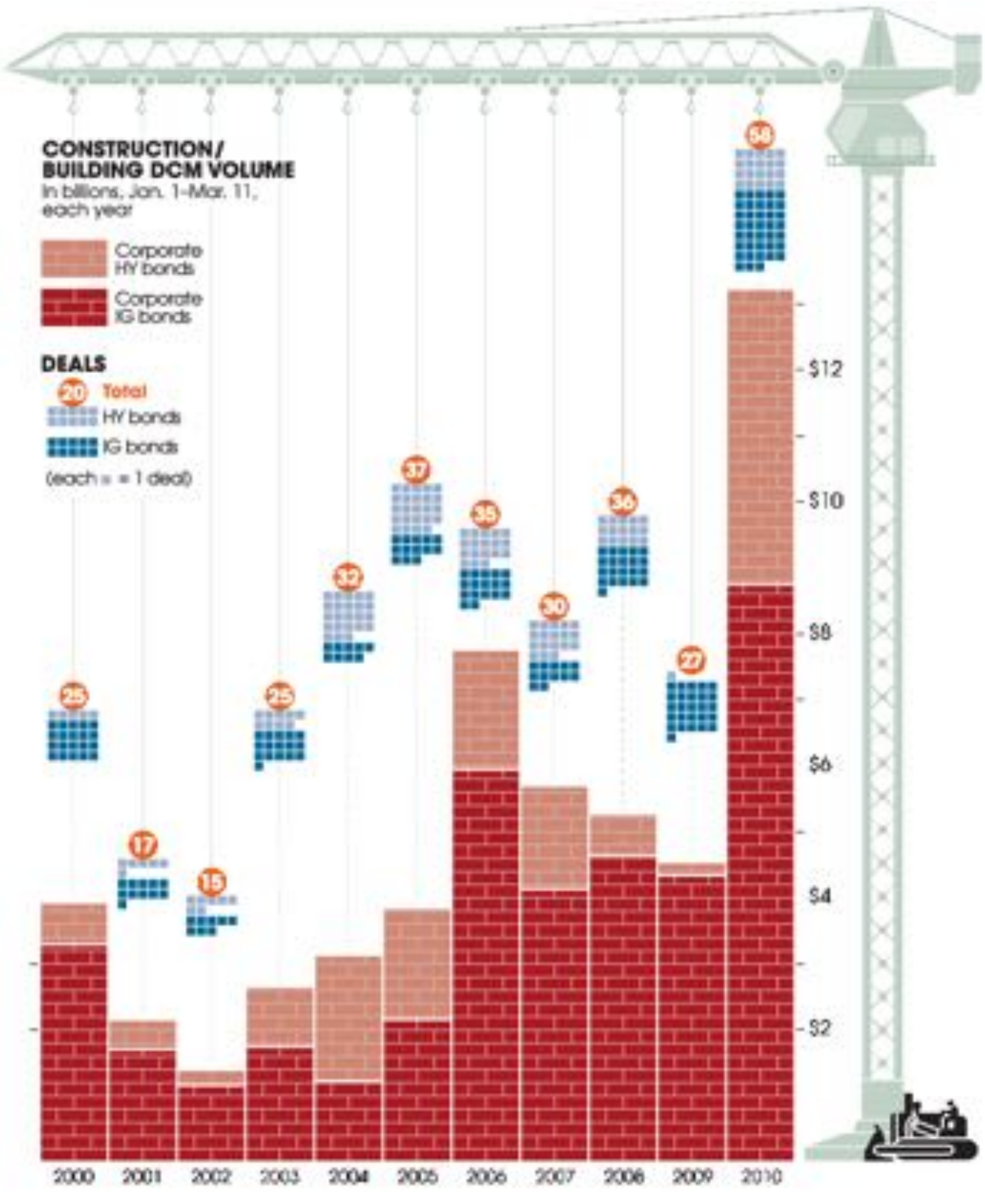

DIAMONDS *WERE* A GIRL'S BEST FRIEND
Average price of a one-carat D-flawless
$60,000
$50,000
$40,000
$30,000
$20,000
1978
1979
1980
1981
1982
TIME Chart by Nigel Holmes
Source: The Diamond Registry

NIGEL HOLMES'S DESIGN PRINCIPLES

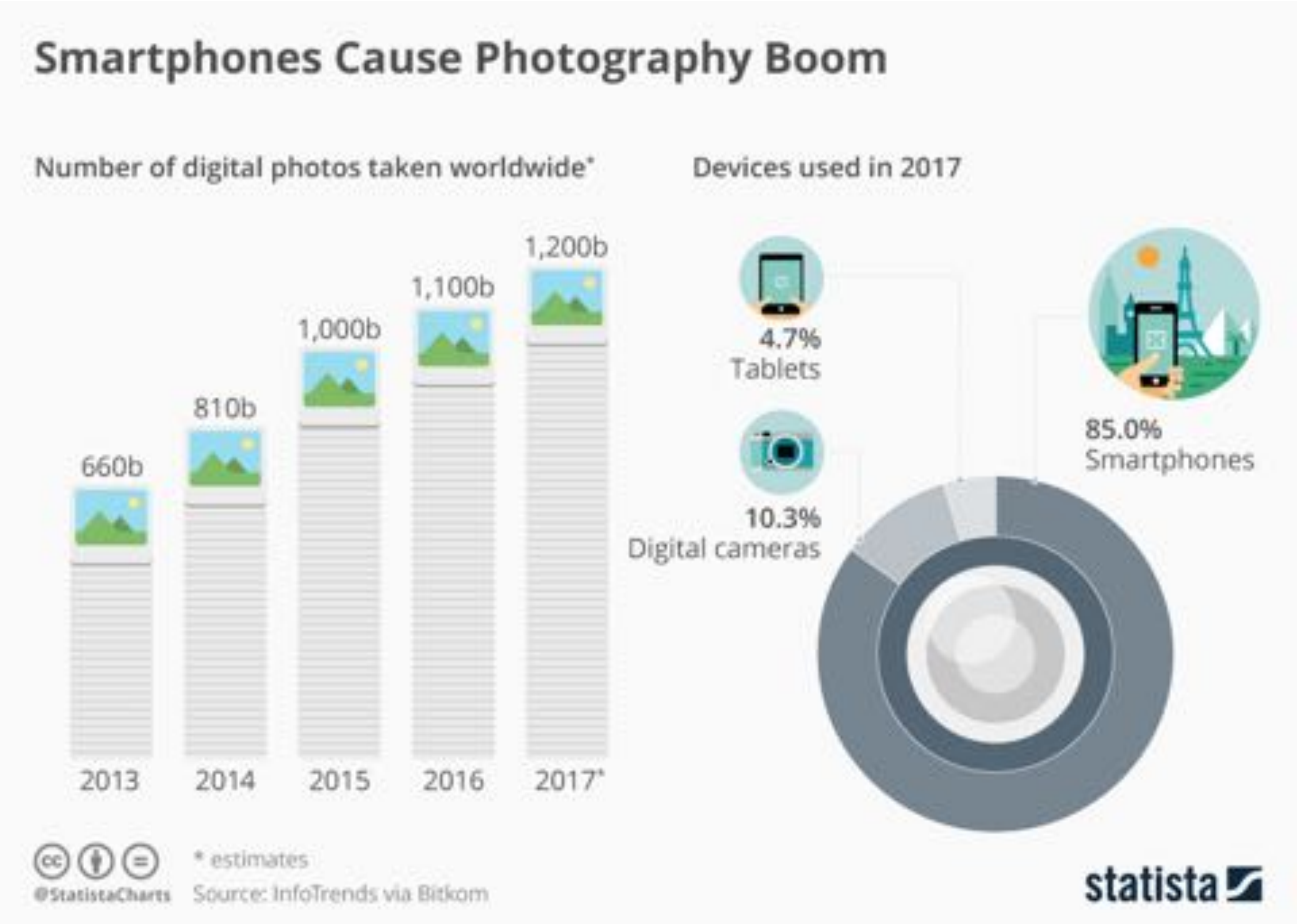
Use humor to instill affection in readers for numbers and charts



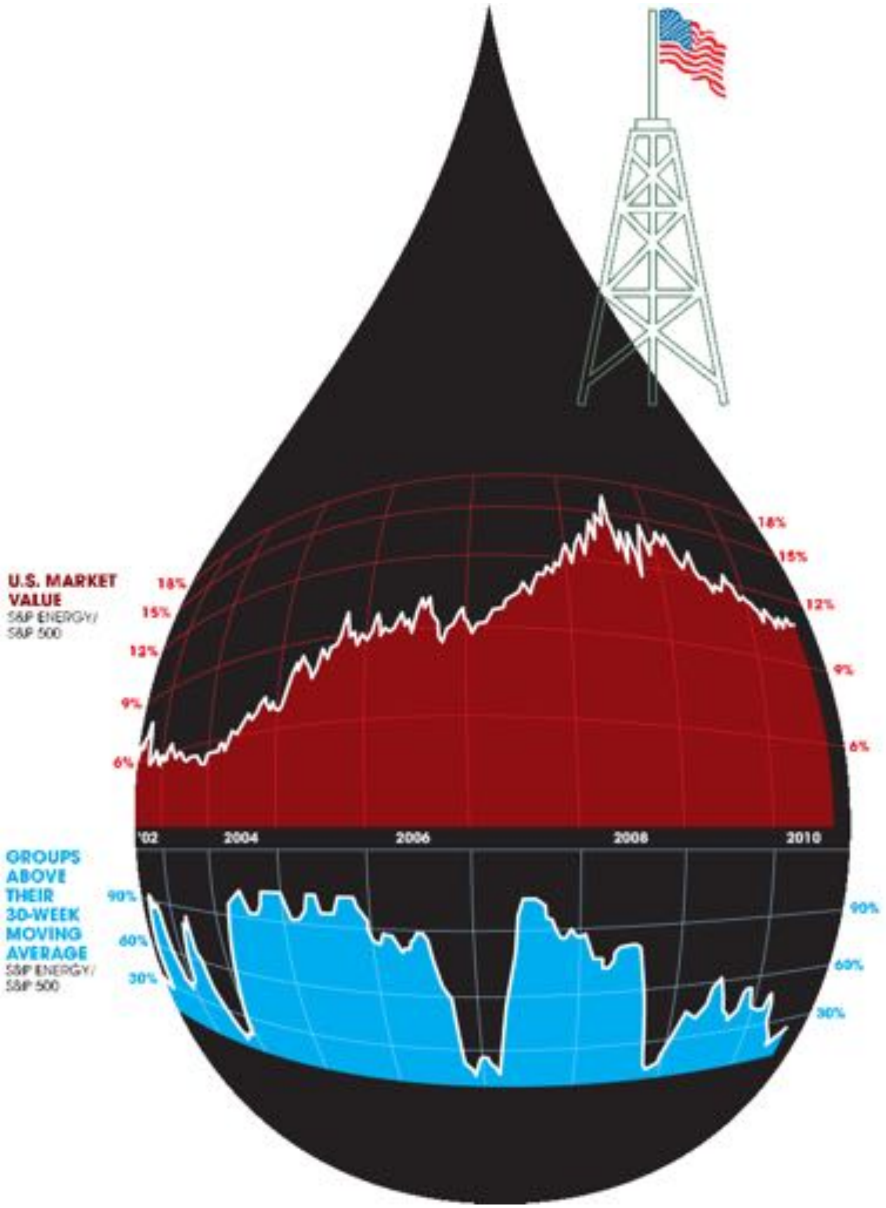
Nigel Holmes Website



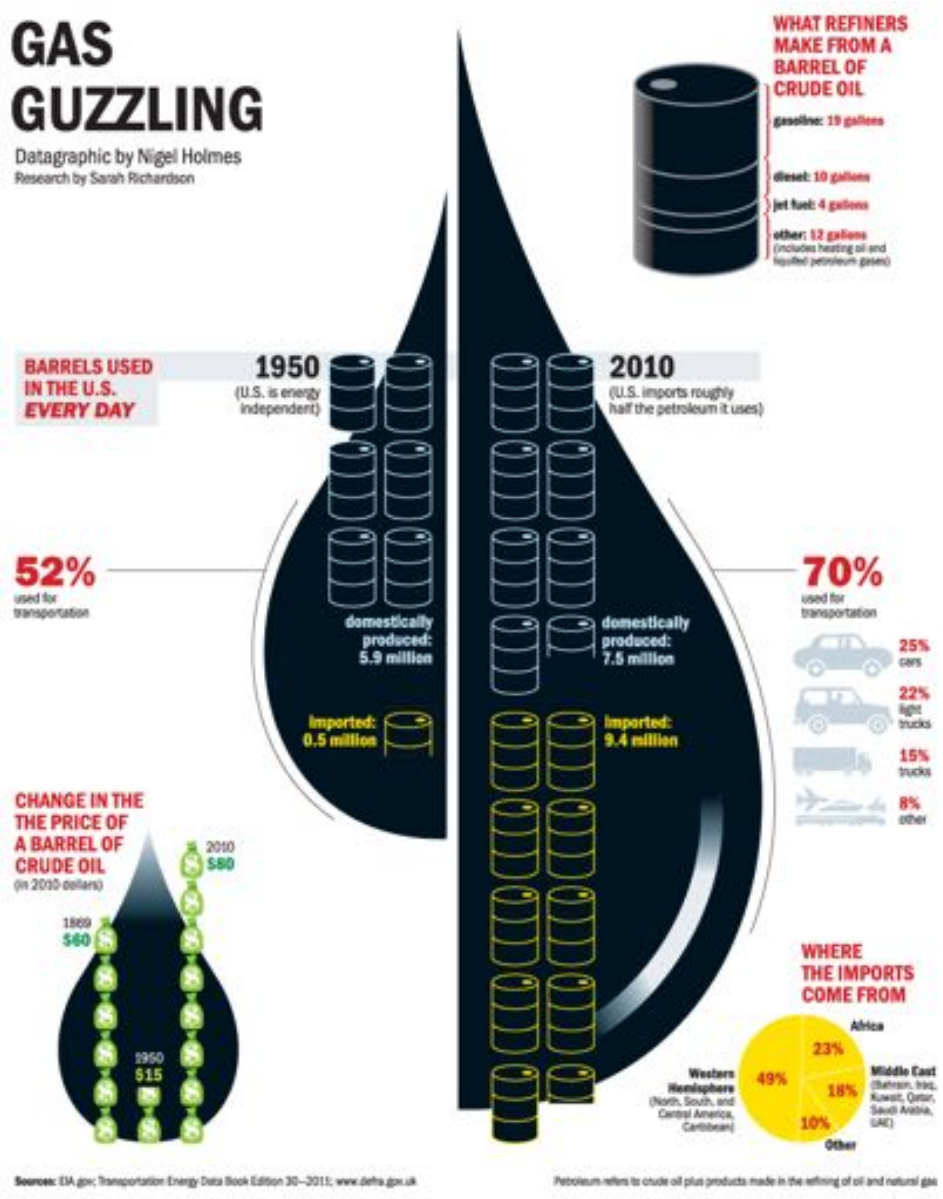
Nigel Holmes Website



Statista - Smartphones Cause Photography Boom



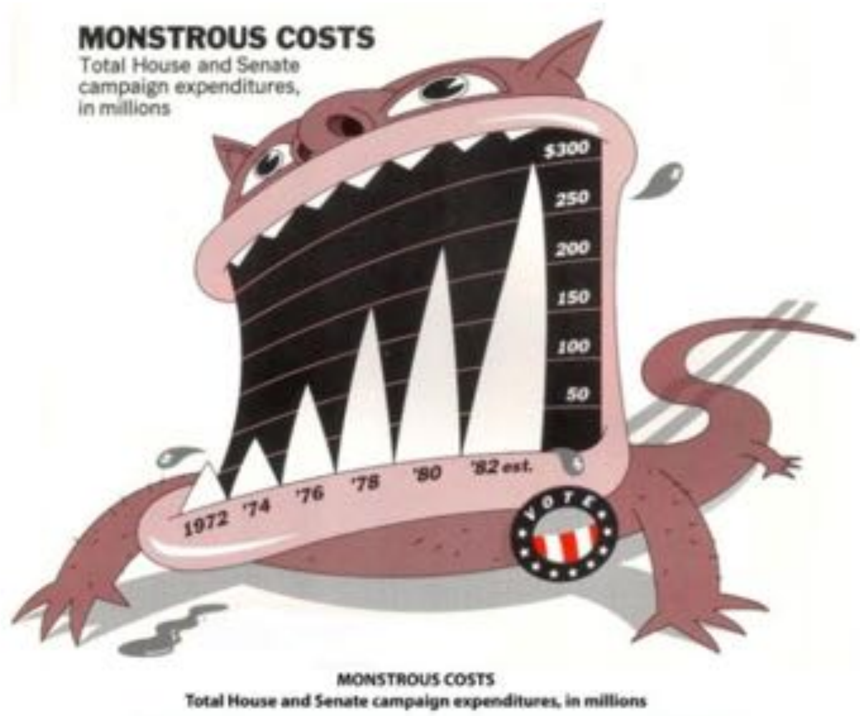
Nigel Holmes Website



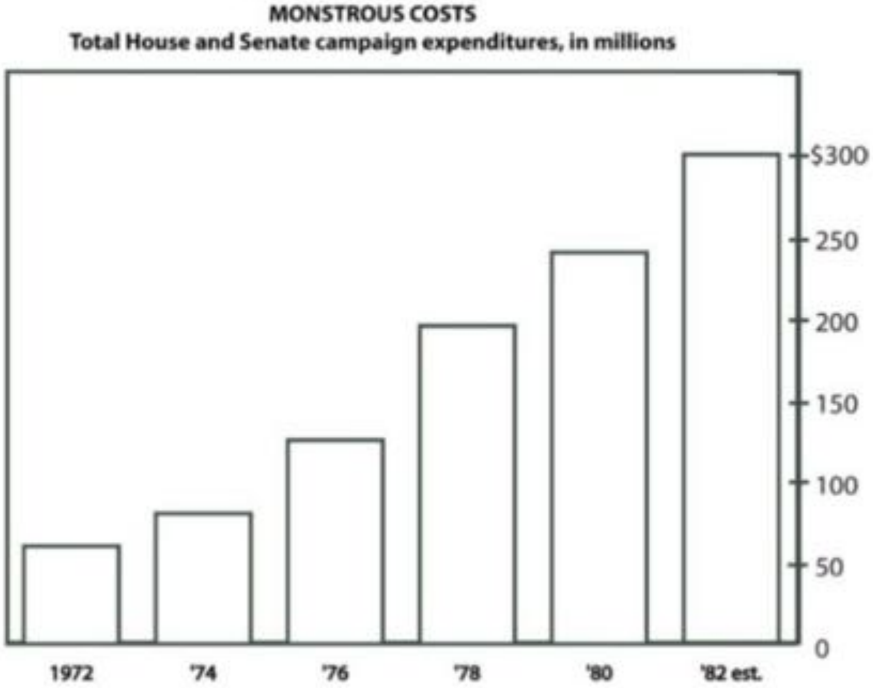
Nigel Holmes Website

USEFUL JUNK? [BATEMAN 2010]

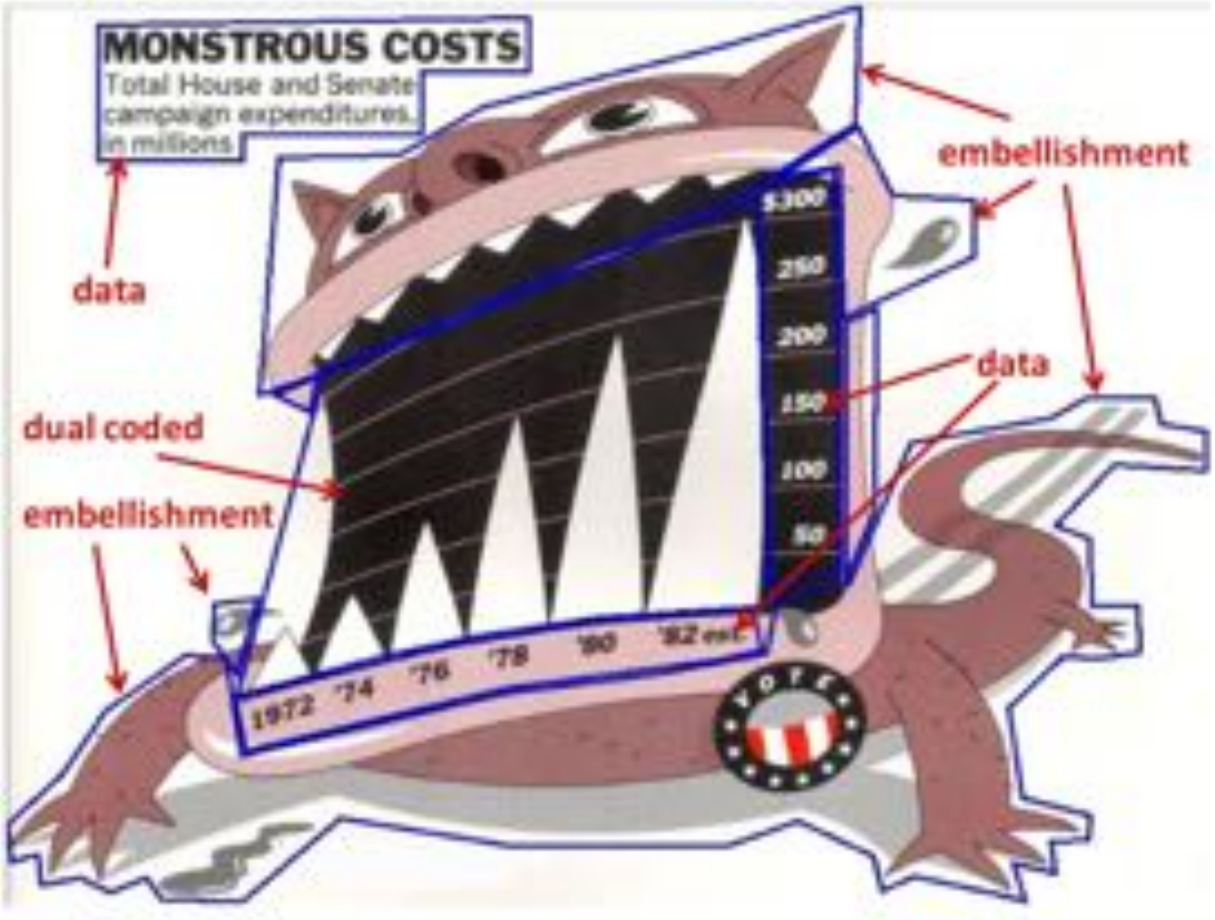
HOLMES



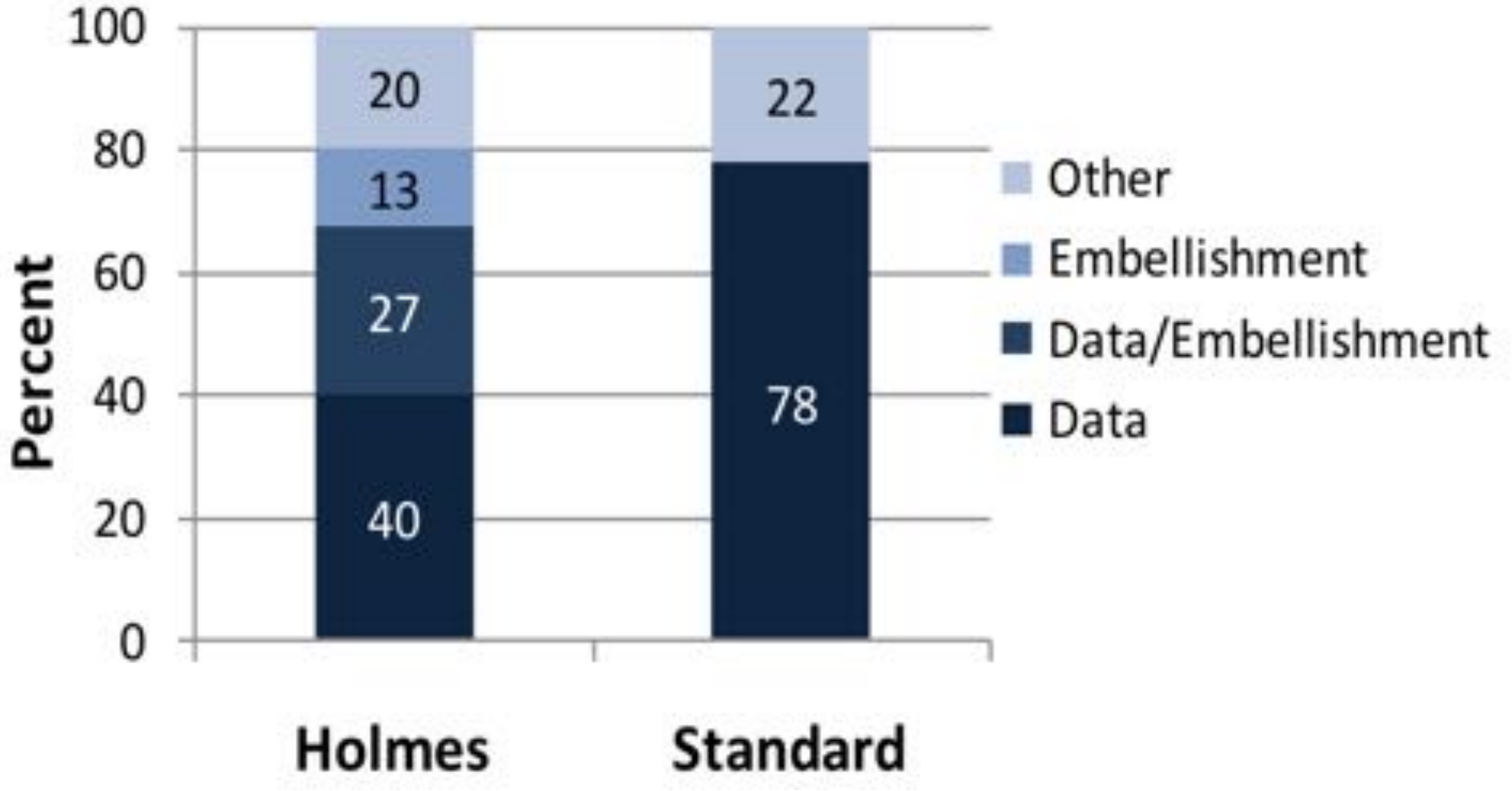
STANDARD



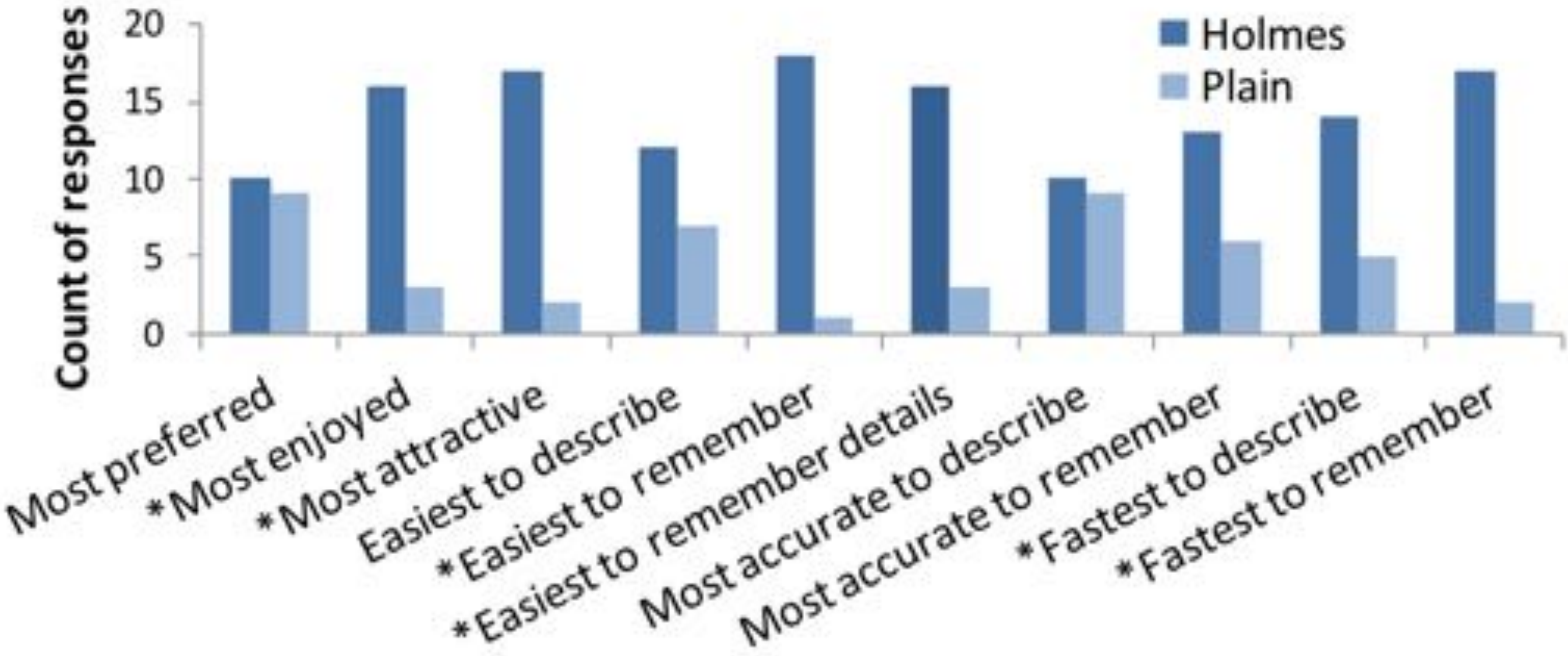
Bateman et al. Useful Junk? The Effects of Visual Embellishment on Comprehension and Memorability of Charts.
ACM Conference on Human Factors in Computing Systems, Atlanta, GA, USA. 2010.



Labeled chart elements



Screen time spent looking at different chart elements



* significantly better for Holmes

No difference:
- Interactive interpretation & accuracy
- Recall accuracy after a five-minute gap

Different:
- Readers value messages in Holmes charts more often than in plain

Comprehension and memorability

OUTLINE

- Function and esthetics, minimalistic visualizations
- D3 data join basics
  - Stress test
  - Customize the selection
- Loading data in D3

Javascript

Initial DOM

Final DOM

```
var dataset = [5, 10, 15];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function (d) {
    return d;
  });
```

[Blank Page]

```
<html>
  <body>
  </body>
</html>
```

Data join: empty initial selection

Javascript

```
var dataset = [5, 10, 15];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function (d) {
    return d;
  });
```

Initial DOM

[Blank Page]

```
<html>
  <body>
  </body>
</html>
```

Final DOM

5

10

15

```
<html>
  <body>
  <p>5</p>
  <p>10</p>
  <p>15</p>
  </body>
</html>
```

Data join: empty initial selection

Javascript

```
var dataset = [5, 10, 15];

//d3.select('body')
d3.selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function (d) {
    return d;
  });
```

Initial DOM

[Blank Page]

```
<html>
  <body>
  </body>
</html>
```

Final DOM

Data join: empty initial selection without parent

Javascript

```
var dataset = [5, 10, 15];

//d3.select('body')
d3.selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function (d) {
    return d;
  });
```

Initial DOM

[Blank Page]

```
<html>
  <body>
  </body>
</html>
```

Final DOM

[Blank Page]

```
<html>
  <body>
  </body>
  <p>5</p>
  <p>10</p>
  <p>15</p>
</html>
```

Data join: empty initial selection without parent

Javascript

Initial DOM

Final DOM

```
var dataset = [5, 10, 15];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function (d) {
    return d;
  });
```

A

B

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
  </body>
</html>
```

Data join: non empty initial selection

Javascript

```
var dataset = [5, 10, 15];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function (d) {
    return d;
  });
```

Initial DOM

A

B

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
  </body>
</html>
```

Final DOM

A

B

15

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
    <p>15</p>
  </body>
</html>
```

Data join: non empty initial selection

Javascript | Initial DOM | Final DOM

```
var dataset = [5, 10, 15];

d3.select('body')
d3.selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function (d) {
    return d;
  });
```

A

B

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
  </body>
</html>
```

Data join: non empty initial selection but no parent

Javascript

```
var dataset = [5, 10, 15];

d3.select('body')
d3.selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function (d) {
    return d;
  });
```

Initial DOM

A

B

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
  </body>
</html>
```

Final DOM

A

B

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
  </body>
  <p>15</p>
</html>
```

Data join: non empty initial selection but no parent

Javascript

```
var dataset = [5, 10, 15];

d3.select('body')
  .selectAll('div')
  .data(dataset)
  .enter()
  .append('span')
  .text(function (d) {
    return d;
  });
```

Initial DOM

A

B

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
  </body>
</html>
```

Final DOM

Data join: selecting and appending different elements

Javascript

```
var dataset = [5, 10, 15];

d3.select('body')
  .selectAll('div')
  .data(dataset)
  .enter()
  .append('span')
  .text(function (d) {
    return d;
  });
```

Initial DOM

A

B

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
  </body>
</html>
```

Final DOM

A

B

5 10 15

```
<html>
  <body>
    <p>A</p>
    <p>B</p>
    <span>5</span>
    <span>10</span>
    <span>15</span>
  </body>
</html>
```

Data join: selecting and appending different elements

OUTLINE

- Function and esthetics, minimalistic visualizations
- D3 data join basics
  - Stress test
  - Customize the selection
  - Multiple elements per data point
- Loading data in D3

CUSTOMIZE THE SELECTION

HTML ELEMENTS

- `.attr()` to set attributes, e.g., `class`
- `.style()` to set style parameters
- `.text()` to set inner text

SVG ELEMENTS

1. `.attr()` to place and size, e.g., `x`, `width`
2. `.style()` to configure and update appearance

OUTLINE

- Function and esthetics, minimalistic visualizations
- D3 data join basics
  - Stress test
  - Customize the selection
  - Multiple elements per data point
- Loading data in D3

SELECTING CONFIGURING THE PARENT

SELECTING

```
var el = d3.select('body')  //select body

var el = d3.select('#div0')  //select div with id div0

var el = d3.select('#svg0')  //select svg with id svg0
```

SIZING SVG

Statically

```
<svg id='svg0' width='300' height='100'></svg>
```

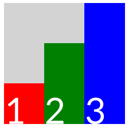
Dynamically

```
var svg = d3.select('body')
   .append('svg')
   .attr('width', '300')
   .attr('height', '100');
```

OUTLINE

- Function and esthetics, minimalistic visualizations
- D3 data join basics
  - Stress test
  - Customize the selection
  - Multiple elements per data point
- Loading data in D3

PROBLEM: MULTIPLE ELEMENTS PER DATA POINT



```
<svg width="60px" height="60px" style="background-color: lightgrey">
  <!-- First data point -->
  <rect x="0" y="40" width="20" height="20" fill="red"></rect>
  <text x="0" y="60" font-size="24" fill="white">1</text>

  <!-- Second data point -->
  <rect x="20" y="20" width="20" height="40" fill="green"></rect>
  <text x="20" y="60" font-size="24" fill="white">2</text>

  <!-- Third data point -->
  <rect x="40" y="0" width="20" height="60" fill="blue"></rect>
  <text x="40" y="60" font-size="24px" fill="white">3</text>
</svg>
```

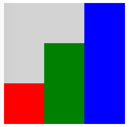
⚠️ D3 BARS DATA JOIN (THE WRONG WAY)

```
<body>
<svg width="60px" height="60px" style="background-color: lightgrey" id="chart1"></svg>
<script>
var dataset = [{name: '1', color: 'red', width: 20, height: 20},
               {name: '2', color: 'green', width: 20, height: 40},
               {name: '3', color: 'blue', width: 20, height: 60}];

d3.select('#chart1')
  .selectAll('rect')
  .data(dataset)
  .enter()

  .append('rect')
  .attr('x', function (d, i) { return i * d.width; })
  .attr('y', function (d) { return 60 - d.height; })
  .attr('width', function (d) { return d.width; })
  .attr('height', function (d) { return d.height; })
  .attr('fill', function (d) { return d.color;})

  .append('text')  // NOT SEEN!
  .attr('x', function (d, i) { return i * d.width; })
  .attr('y', function (d) { return 60; })
  .attr('font-size', '18px')
  .attr('fill', 'white')
  .text(function (d) { return d.name; })
</script>
</body>
```



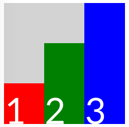
👍 D3 BARS DATA JOIN (2 DATA JOINS)

```
<body>
<svg width="60px" height="60px" style="background-color: lightgrey" id="chart2"></svg>
<script>
var dataset = [{name: '1', color: 'red', width: 20, height: 20},
               {name: '2', color: 'green', width: 20, height: 40},
               {name: '3', color: 'blue', width: 20, height: 60}];

var svg = d3.select('#chart2');

svg.selectAll('rect')
  .data(dataset)
  .enter()
  .append('rect')
  .attr('x', function (d, i) { return i * d.width; })
  .attr('y', function (d) { return 60 - d.height; })
  .attr('width', function (d) { return d.width; })
  .attr('height', function (d) { return d.height; })
  .attr('fill', function (d) { return d.color;});

svg.selectAll('text')
  .data(dataset)
  .enter()
  .append('text')
  .attr('x', function (d, i) { return i * d.width; })
  .attr('y', function (d) { return 60; })
  .attr('font-size', '18px')
  .attr('fill', 'white')
  .text(function (d) { return d.name; })
</script>
</body>
```



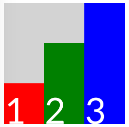
👍 D3 BARS DATA JOIN (ENTER SELECTION)

```
<body>
<svg width="60px" height="60px" style="background-color: lightgrey" id="chart2"></svg>
<script>
var dataset = [{name: '1', color: 'red', width: 20, height: 20},
               {name: '2', color: 'green', width: 20, height: 40},
               {name: '3', color: 'blue', width: 20, height: 60}];

var enter_selection = d3.select('#chart2')
  .selectAll('rect')
  .data(dataset)
  .enter();

enter_selection.append('rect')
  .attr('x', function (d, i) { return i * d.width; })
  .attr('y', function (d) { return 60 - d.height; })
  .attr('width', function (d) { return d.width; })
  .attr('height', function (d) { return d.height; })
  .attr('fill', function (d) { return d.color;});

enter_selection.append('text')
  .attr('x', function (d, i) { return i * d.width; })
  .attr('y', function (d) { return 60; })
  .attr('font-size', '18px')
  .attr('fill', 'white')
  .text(function (d) { return d.name; })
</script>
</body>
```



OUTLINE

- Function and esthetics, minimalistic visualizations
- D3 data join basics
  - Stress test
  - Customize the selection
  - Multiple elements per data point
- Loading data in D3

LOADING DATA IN D3

- Browsers cannot access or load local data (security)!
- Browsers can load data files by issuing an http request to a server
- Asynchronous process:
  - AJAX calls
  - d3.csv(), d3.json() (callbacks up to v3, now promises)
- Depending on the format Javascript dynamic typing not sufficient

ES6 PROMISES

A promise allows to associate handlers with asynchronous actions

```
let myFirstPromise = new Promise((resolve, reject) => {
  // We call resolve(...) when what we were doing asynchronously was successful, and reject(...) when it failed.
  // In this example, we use setTimeout(...) to simulate async code.
  setTimeout( function() {
    resolve("Success!")  // Yay! Everything went well!
  }, 250)
})

myFirstPromise.then((successMessage) => {
  // successMessage is whatever we passed in the resolve(...) function above.
  // It doesn't have to be a string, but if it is only a succeed message, it probably will be.
  console.log("Yay! " + successMessage)
});
```

LOADING CSV FILES

```
$ cat > cars.csv
Year,Make,Model,Length
1997,Ford,E350,2.34
2000,Mercury,Cougar,2.38
```

```
d3.csv("cars.csv").then(function (data) {
  console.log(data);
});
```

Output in terminal

```
[{Year: "1997", Make: "Ford", Model: "E350", Length: "2.34"},
 {Year: "2000", Make: "Mercury", Model: "Cougar", Length: "2.38"}]
```

CONVERTING TO NUMBERS

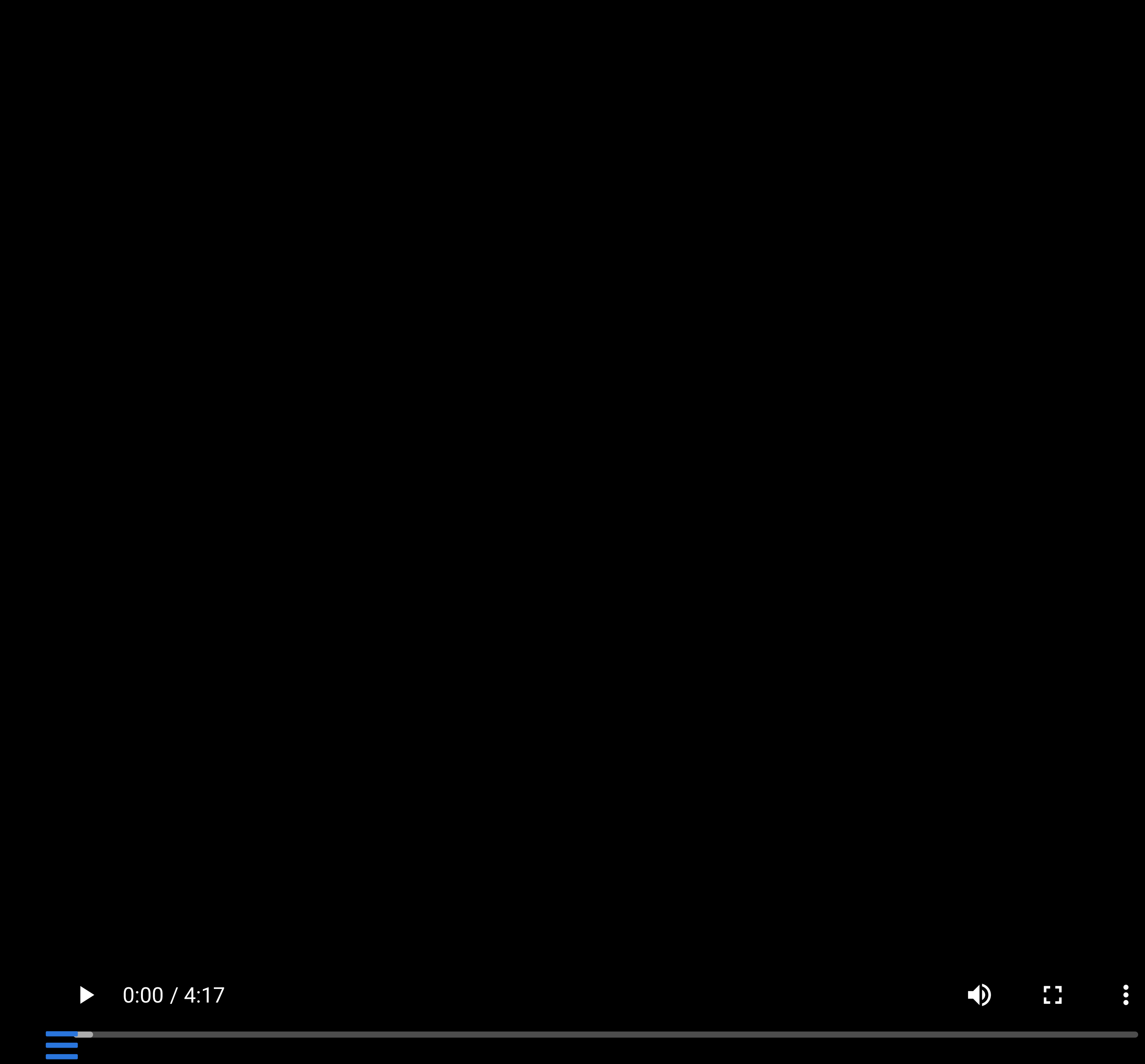
`parseInt()` and `parseFloat()`

```
parseInt('10');  //int 10
parseFloat('10.1');  //float 10.1
```

Coercion with “*unary + operator*” (faster)

```
+''  //int 0
+'1'  //int 1
+'1.1'  //float 1.1
```

A lightning talk by Gary Bernhardt from CodeMash 2012


0:00 / 4:17

JSON FORMAT

- Stands for Javascript object notation
- Text format
- Data is represented as a Javascript object
- Keys must be quoted (strings)

```
[
  {"year": 1997, "make": "Ford", "model": "E350", "length": 2.34},
  {"year": 2000, "make": "Mercury", "model": "Cougar", "length": 2.38}
]
```

json.org Introducing JSON

LOADING JSON FILES

```
$ cat > cars.json
[{"year": 1997, "make": "Ford", "model": "E350", "length": 2.34},
 {"year": 2000, "make": "Mercury", "model": "Cougar", "length": 2.38}]
```

```
d3.json("cars.json").then(function (data) {

  console.log(data);

  //prints to the console
  //[{year: 1997, make: "Ford", model: "E350", length: 2.34},
  // {year: 2000, make: "Mercury", model: "Cougar", length: 2.38}]
});
```

d3 API: d3-fetch d3.json(input[, init])

What will appear on the page?

```
Color
Red
Green
Blue
```

data.csv

```
<p>Orange</p>
<script>
d3.csv("data.csv").then(function(data) {
  d3.select("body")
    .selectAll("p")
    .data(data)
    .enter()
    .append("p")
    .text(function(d) {return d.Color; })
});
</script>
```

A. Orange, Color, Red, Green, Blue on separate lines
B. Orange, Green, Blue on separate lines
C. Color, Red, Green, Blue on separate lines
D. Red, Green, Blue on separate lines

What will appear on the page?

```
Color
Red
Green
Blue
```

data.csv

```
<p>Orange</p>
<script>
d3.csv("data.csv").then(function(data) {
  d3.select("body")
    .selectAll("p")
    .data(data)
    .enter()
    .append("p")
    .text(function(d) {return d.Color; })
});
</script>
```

A. Orange, Color, Red, Green, Blue on separate lines

B. Orange, Green, Blue on separate lines←

C. Color, Red, Green, Blue on separate lines

D. Red, Green, Blue on separate lines