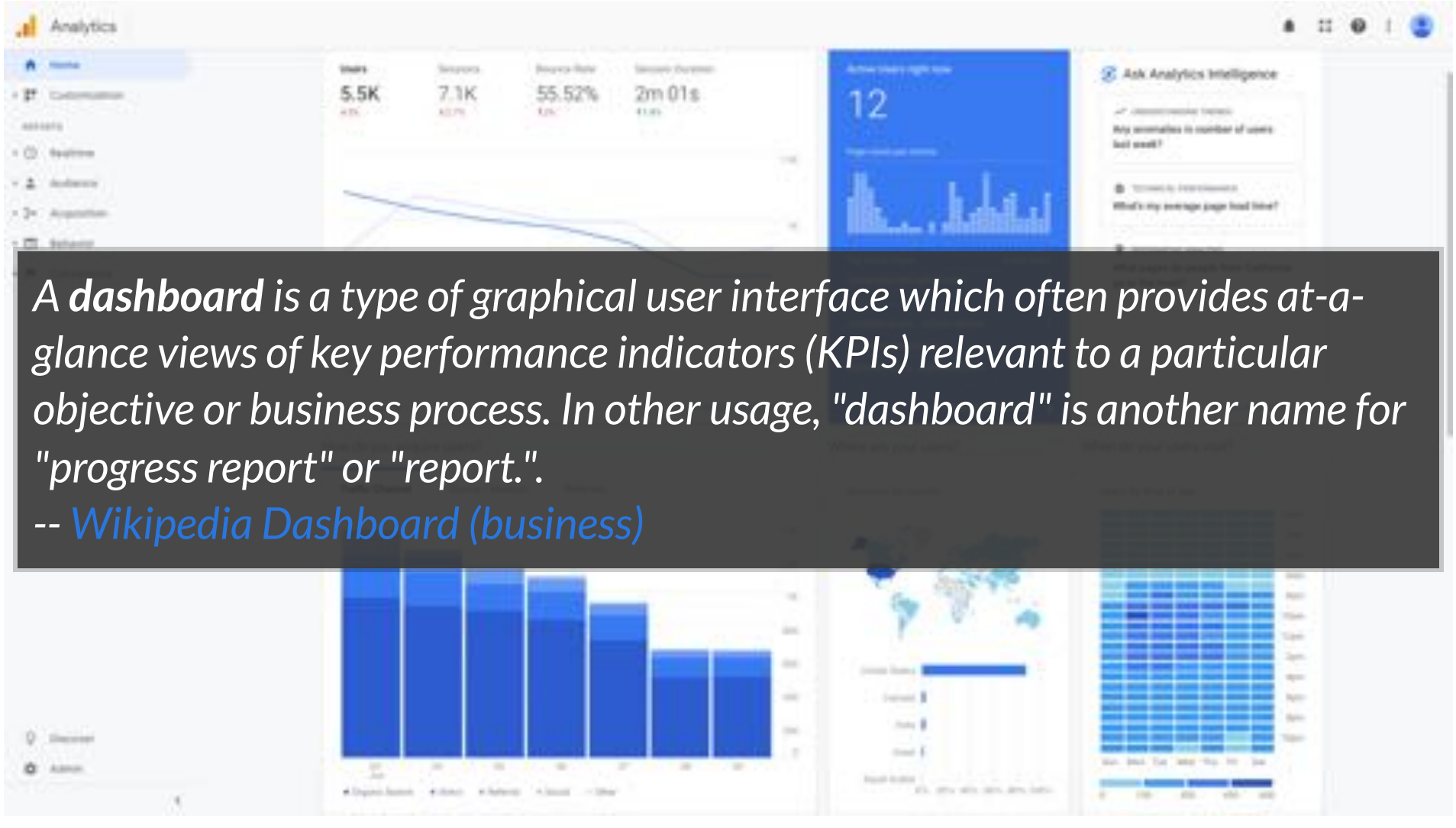
DSCI 554 LECTURE

DASHBOARDS & INFOGRAPHICS DESIGN
D3 DATA JOIN BASICS AND LOADING DATA

Dr. Luciano Nocera

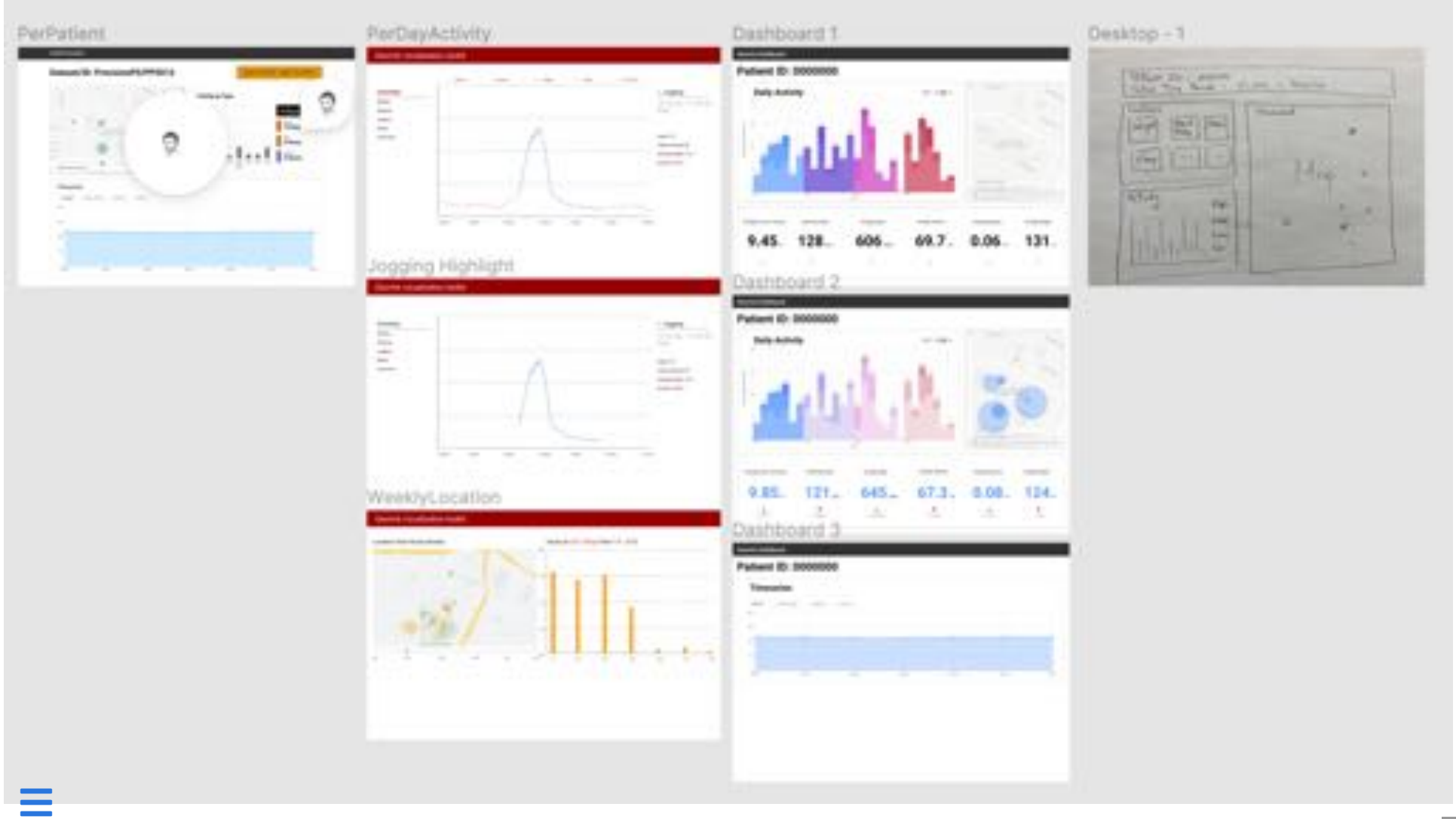
OUTLINE

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

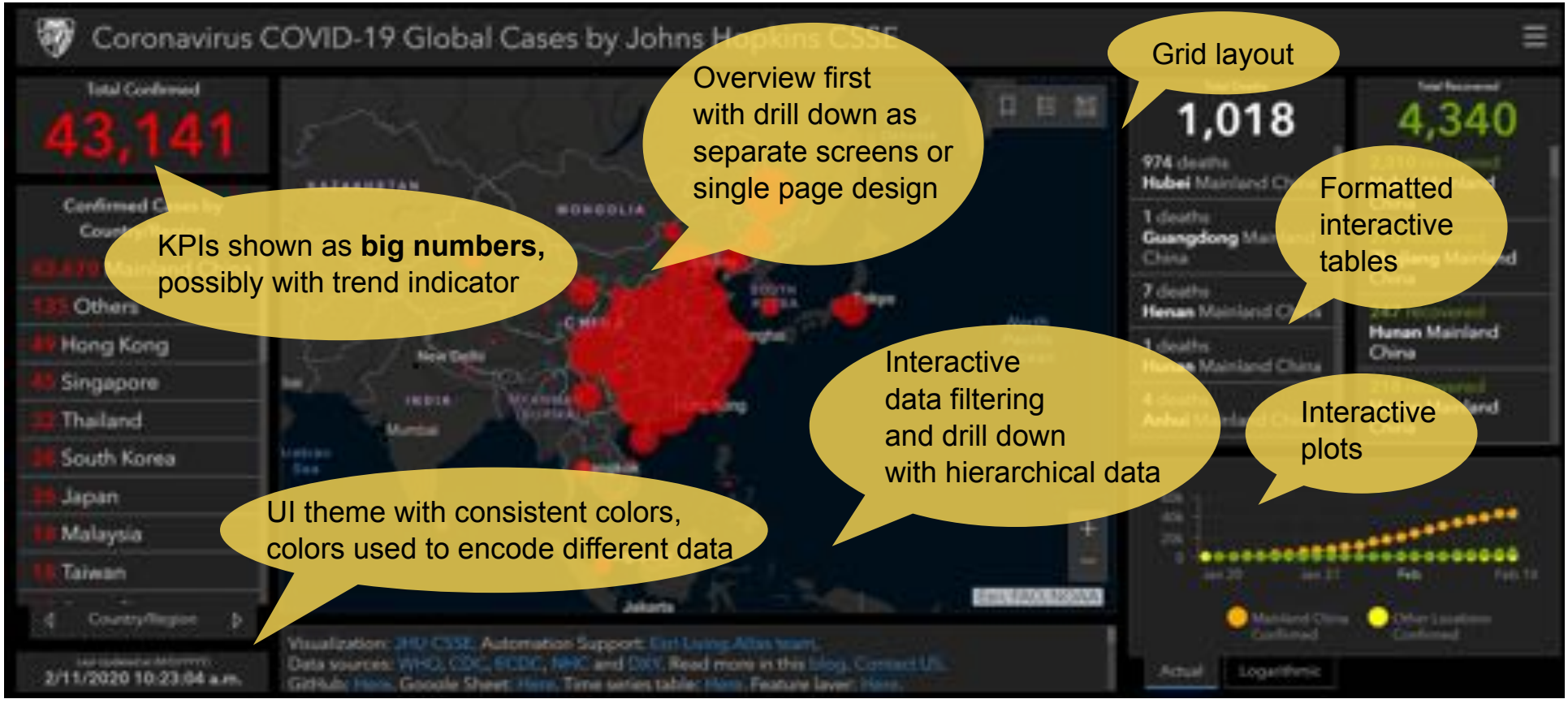


*A **dashboard** is a type of graphical user interface which often provides at-a-glance views of key performance indicators (KPIs) relevant to a particular objective or business process. In other usage, "dashboard" is another name for "progress report" or "report.".*
-- Wikipedia Dashboard (business)

IMPORTANCE OF WIREFRAMING



DASHBOARDS DESIGN ELEMENTS



Coronavirus COVID-19 Global Cases by Johns Hopkins

DASHBOARDS DESIGN

UNDERSTAND

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

APPLY DASHBOARD UX DESIGN PRINCIPLES

- Use a grid layout
- Most important at the top
- Use big numbers for KPIs
- **Validate with the "5-second rule"**: users can identify in 5 seconds or less:
  - Topic
  - Key trends
  - How to explore
- Use colors well for theme and data
- Use **web-safe** fonts

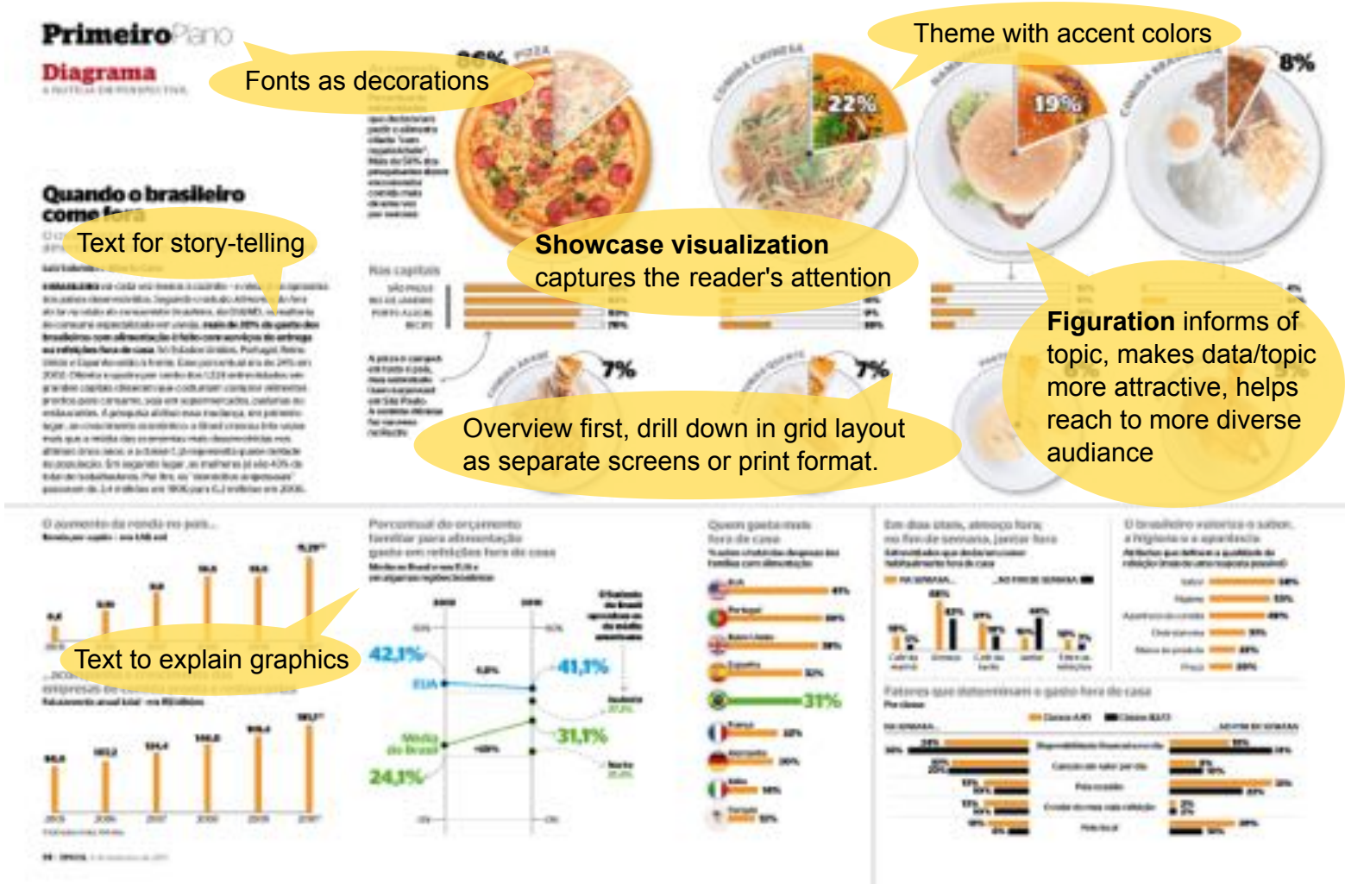
WEB-SAFE FONTS

Web-safe fonts adapt browsers and devices.

Types of web fonts:

- **Serif fonts** contain serifs — small decorative strokes that protrude from the main body of the letter.
- **Sans-Serif fonts** are fonts do not have serifs.
- **`Monospace fonts`** `are fonts that have equal spacing between characters (e.g., Courier).`
- ***Cursive fonts*** *resemble handwriting.*
- **Fantasy fonts** are stylized decorative fonts.

Helvetica (sans-serif)
Arial (sans-serif)
Arial Black (sans-serif)
Verdana (sans-serif)
Tahoma (sans-serif)
Trebuchet MS (sans-serif)
Impact (sans-serif)
Gill Sans (sans-serif)
Times New Roman (serif)
Georgia (serif)
Palatino (serif)
Baskerville (serif)
Andalé Mono (monospace)
Courier (monospace)
Lucida (monospace)
Monaco (monospace)
Bradley Hand (cursive)
Brush Script MT (cursive)
Luminari (fantasy)
Comic Sans MS (cursive)


PrimeiroPlano
Diagrama
Fonts as decorations
Theme with accent colors
Quando o brasileiro come fora
Text for story-telling
Showcase visualization
captures the reader's attention
Figuration informs of topic, makes data/topic more attractive, helps reach to more diverse audiance
Overview first, drill down in grid layout as separate screens or print format.
Text to explain graphics
22%
19%
8%
7%
7%
42,1%
41,1%
31,1%
24,1%
31%

INFOGRAPHIC DESIGN

UNDERSTAND

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

FIND *"SOFT SPOT"* BY ACHIEVING BALANCE [CAIRO]

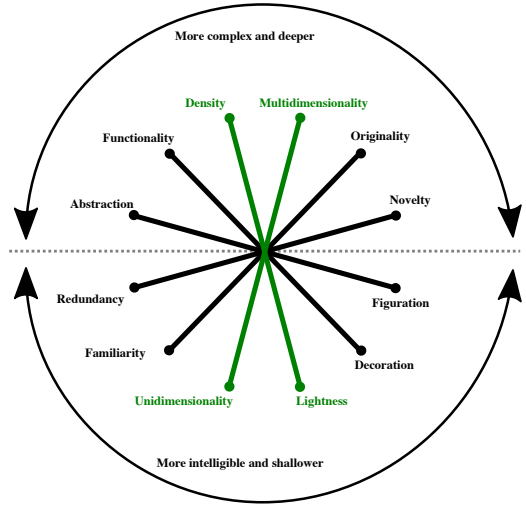
1. **Seek depth:**
   - *"First use the space to help users understand the data, then decorate with a purpose!"*
   - *"Beauty is not the goal of visualization and it is usually not required to achieve the goal… remember that the goal is to enlighten."*
   - *"Do not underestimate users and cater to 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"*

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

1. SEEK DEPTH

Related dimensions:

- Lightness - Density
- Unidimensionality - Multidimensionality



CAIRO'S RECOMMENDATIONS

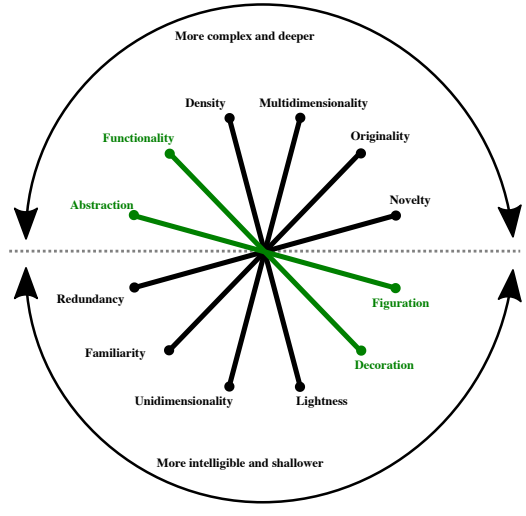
1. Define where your graphic stands in terms of density and dimensionality
2. Move position of graphic at least 10% towards density and multidimensionality
3. Organize in layers, starting with a summary
4. Include inner layers as necessary based on story and focus
5. Structure the layers in logical order

infographic only

2. CLARIFY

Related dimensions:

- Decoration - Functionality
- Figuration - Abstraction



CAIRO'S RECOMMENDATIONS

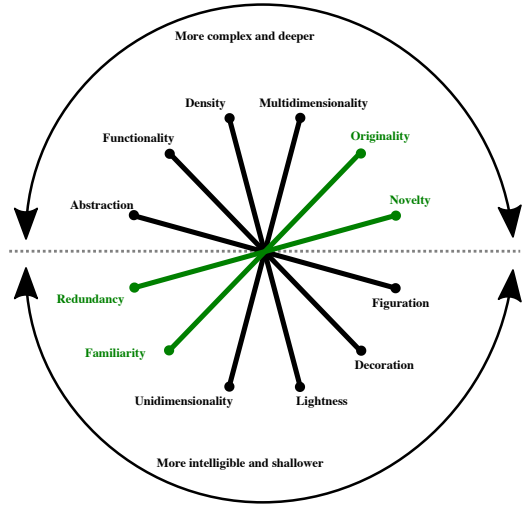
- Do not simplify but clarify
- Think about structure first then eye-candy
- Use space first to explain and develop the story
- Think about how data should be organized before thinking about style
- Never dumb down your data

infographic only

3. BOOM EFFECT

Related dimensions:

- Redundancy - Novelty
- Familiarity - Originality



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

OUTLINE

- Designing dashboards and infographics
- 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 DEFINITIONS

$$\frac{\text{Data-ink}}{\text{Total ink used to print the graphic}}$$

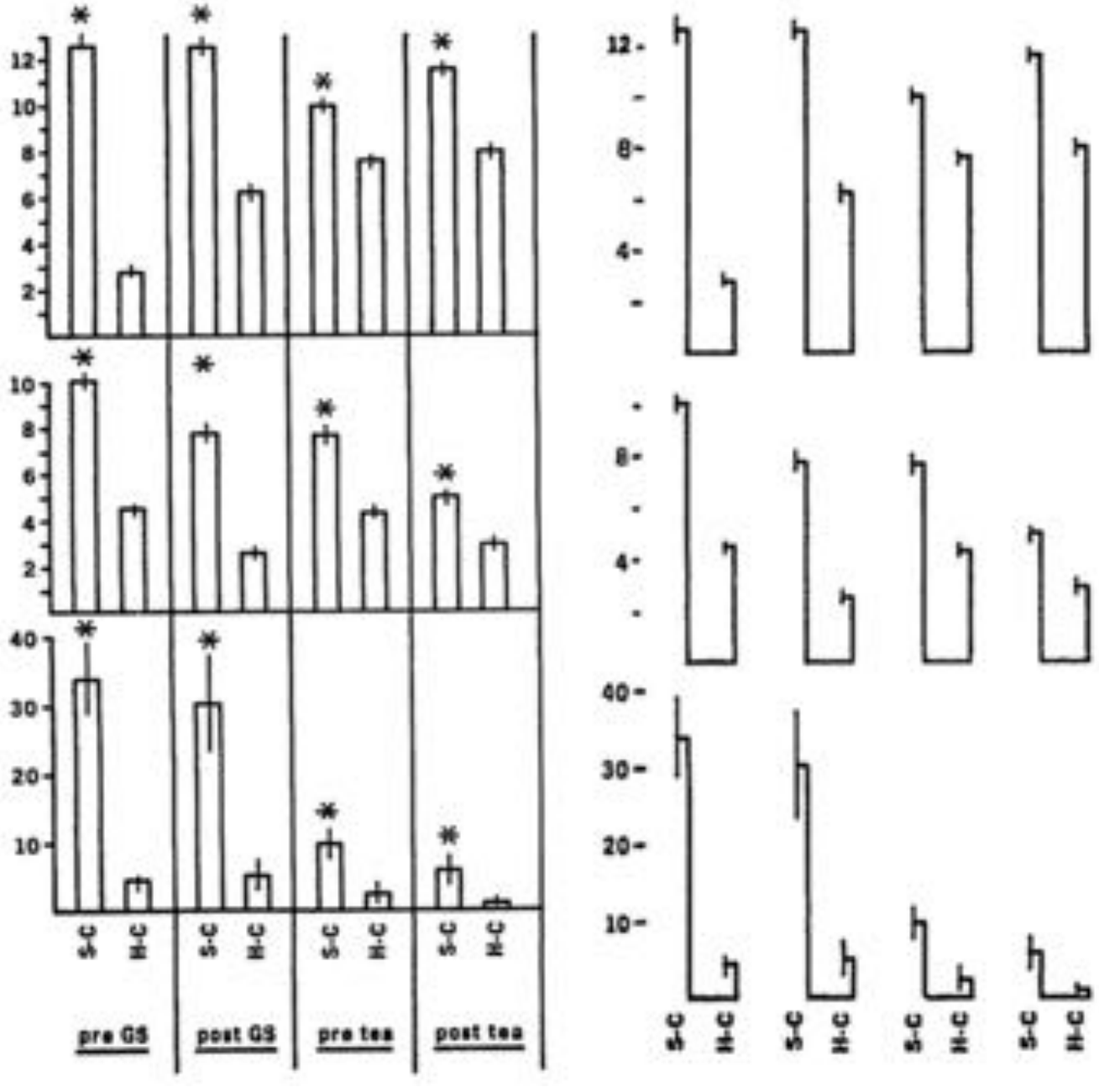
Proportion of a graphic's ink devoted to the non-redundant display of data-information

1.0 – Proportion of a graphic that can be erased without loss of data-information

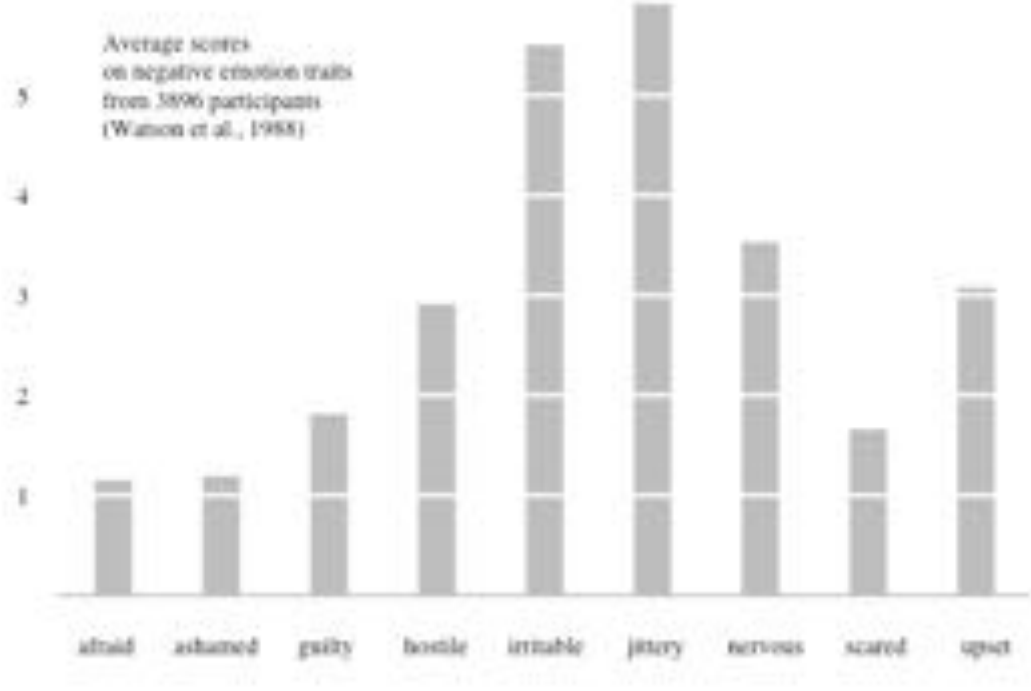
TUFTE'S MINIMALISTIC DESIGN METHOD

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

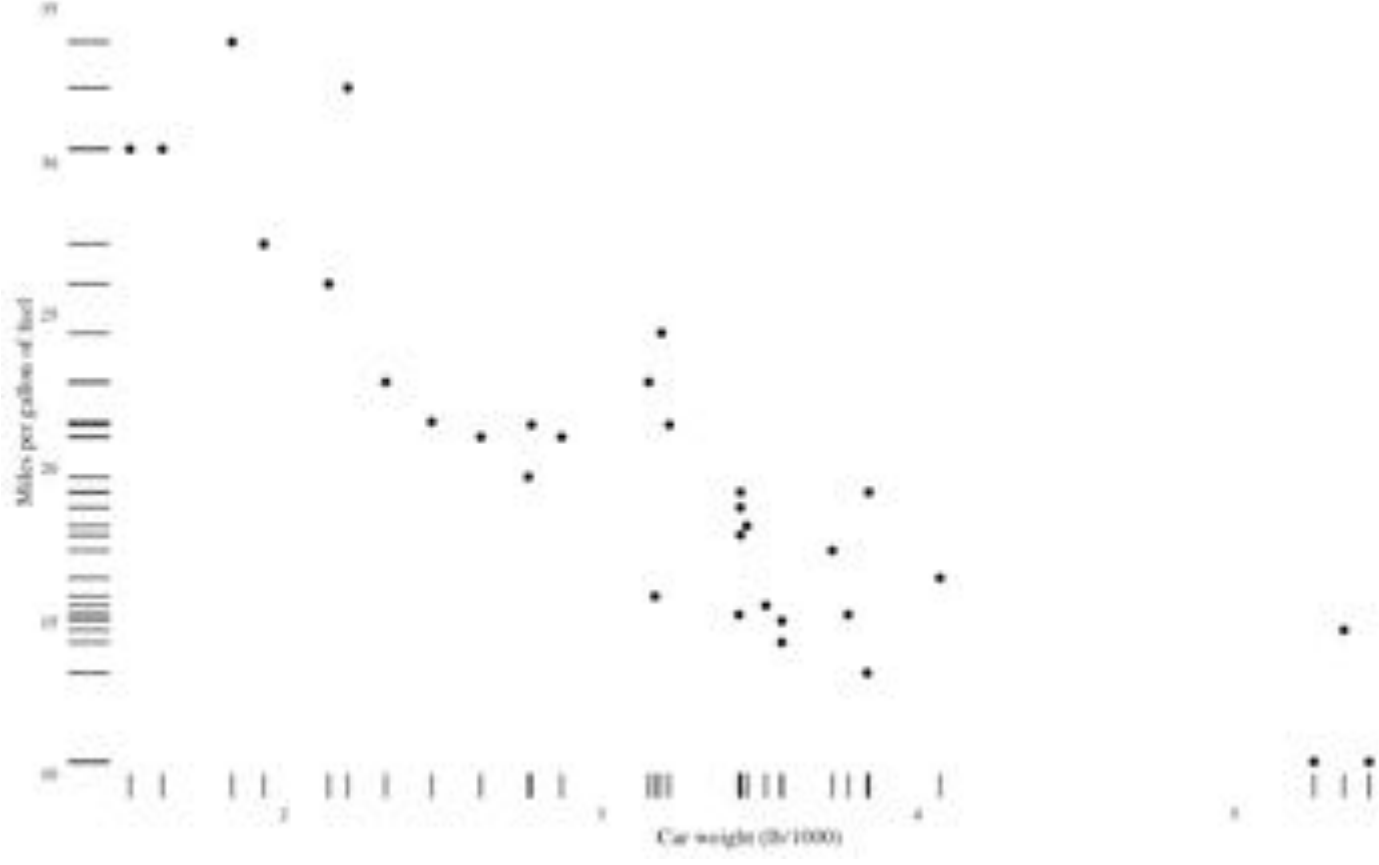
The Visual Display of Quantitative Information, E. Tufte, page 68



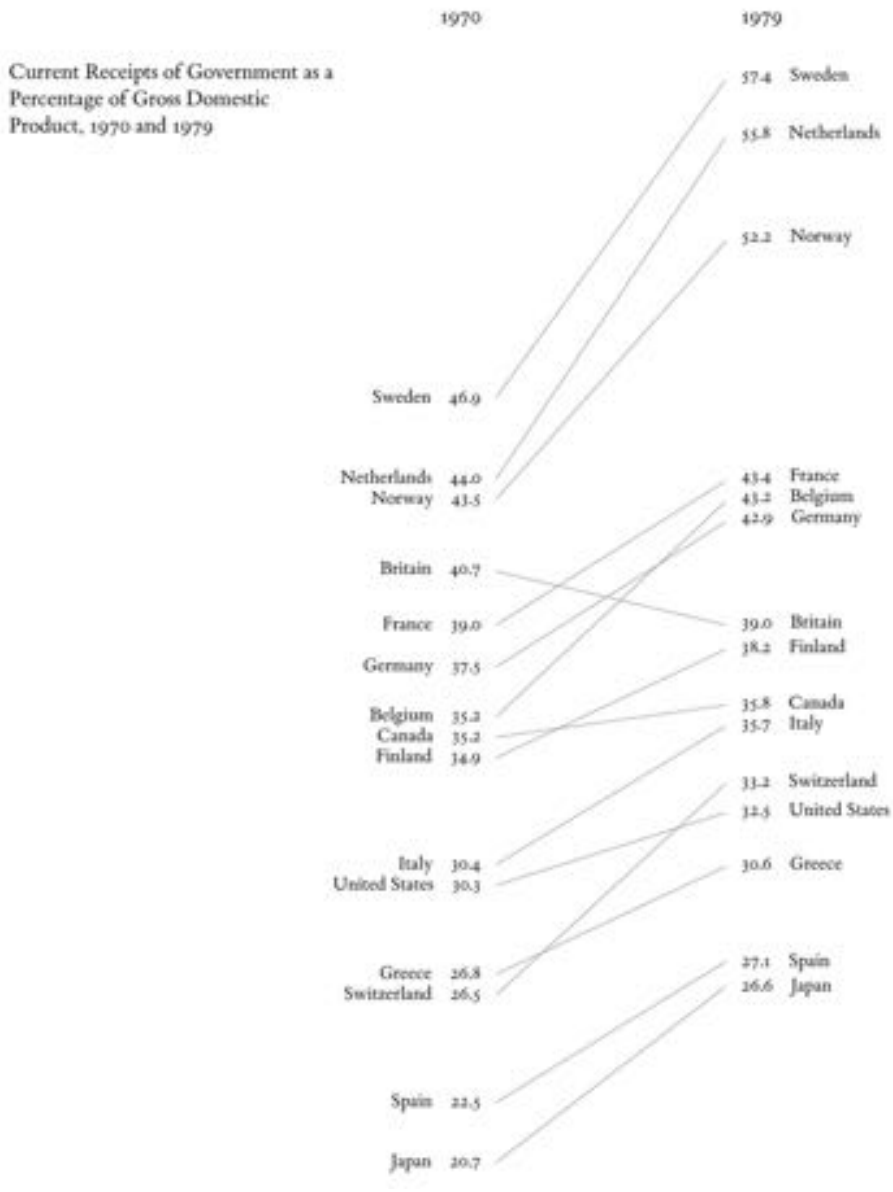
The Visual Display of Quantitative Information, E. Tufte, page 102



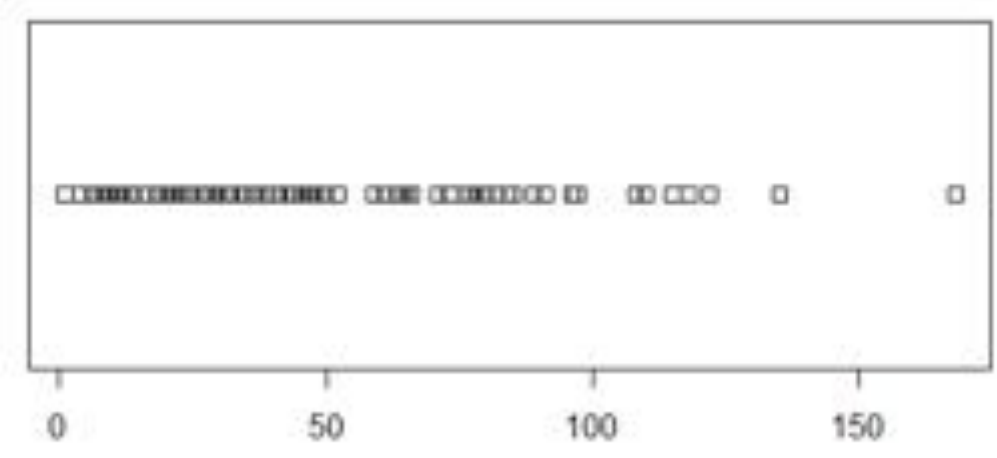
The Visual Display of Quantitative Information" p. 128



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

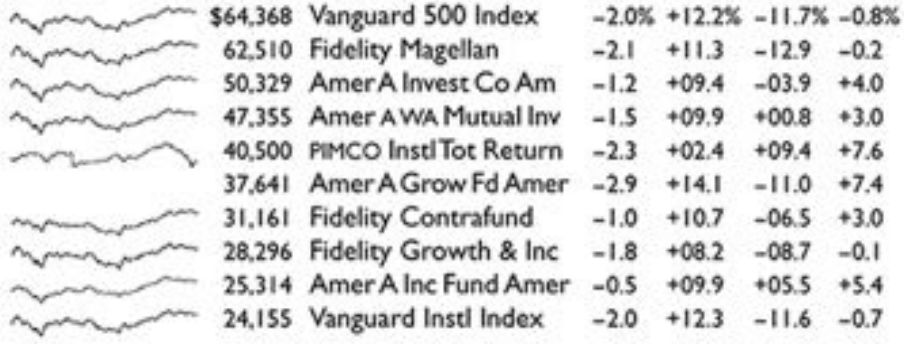


Slopegraph: The Visual Display of Quantitative Information" p. 158



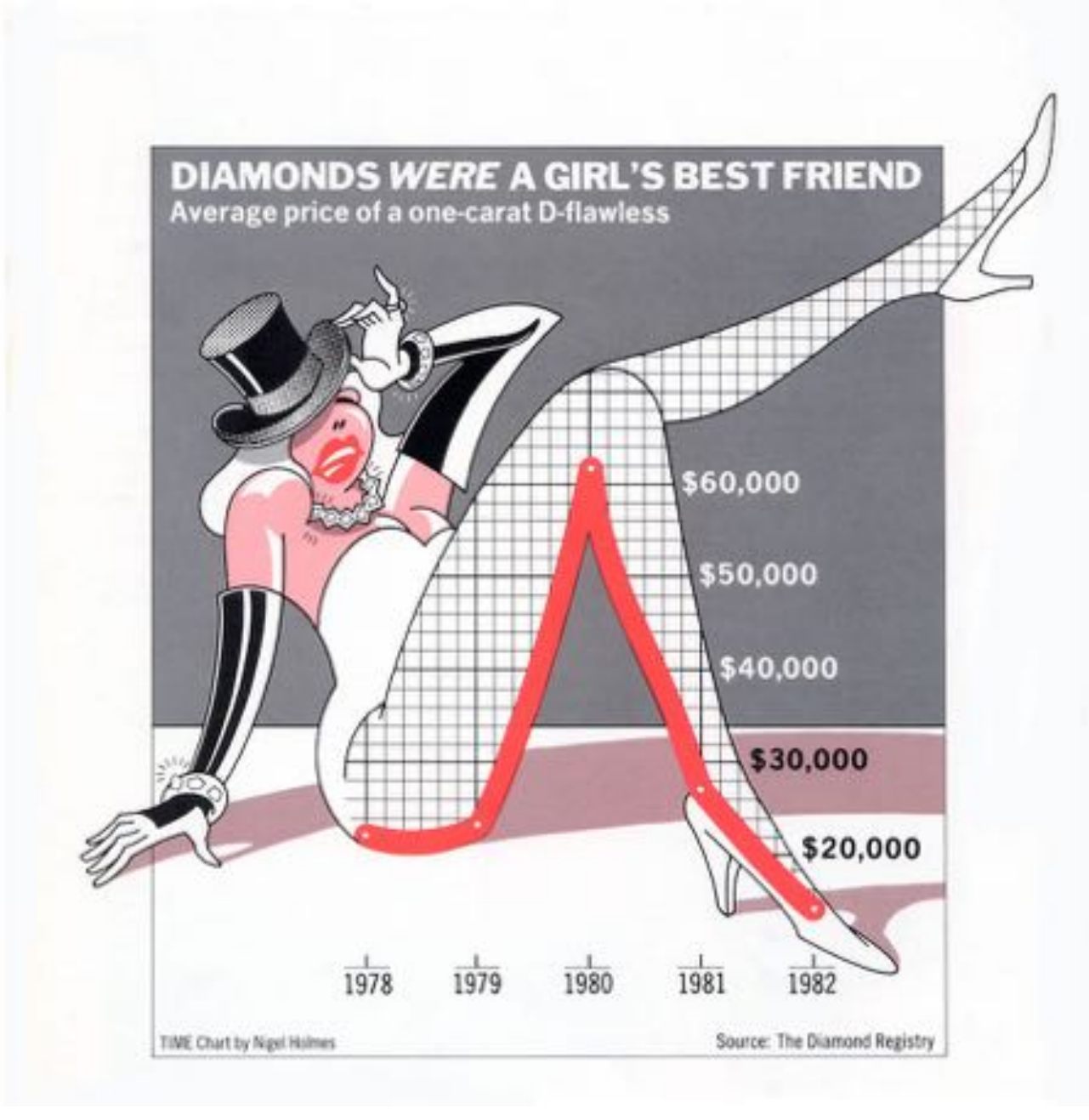
Stripchart: 1-D scatter plot. Good alternative to boxplots when sample sizes are small.

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


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

CHARTJUNK

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

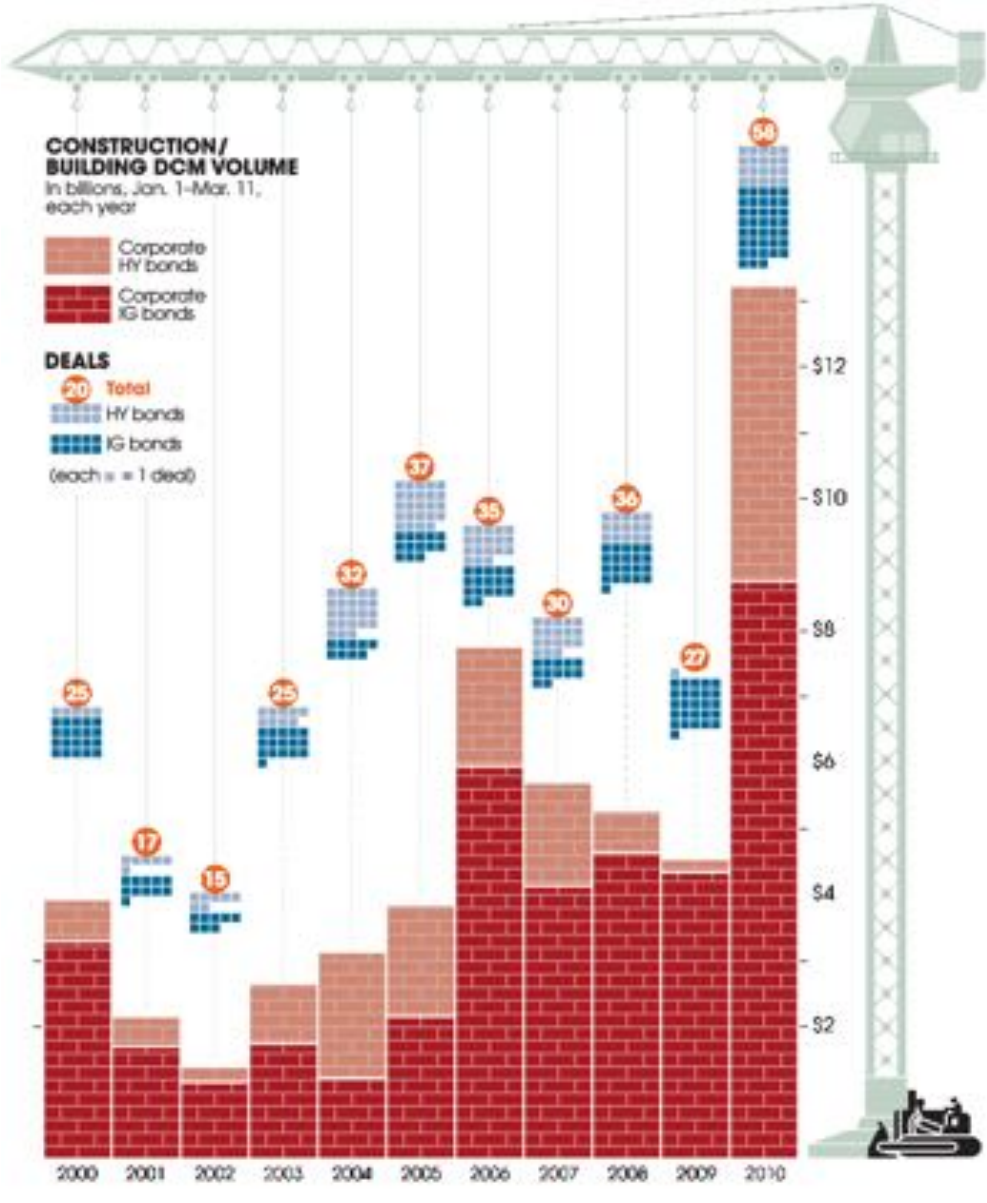
The purpose of decoration varies — to make the graphic appear more scientific and precise, to enliven the display, to give the designer an opportunity to exercise artistic skills.

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

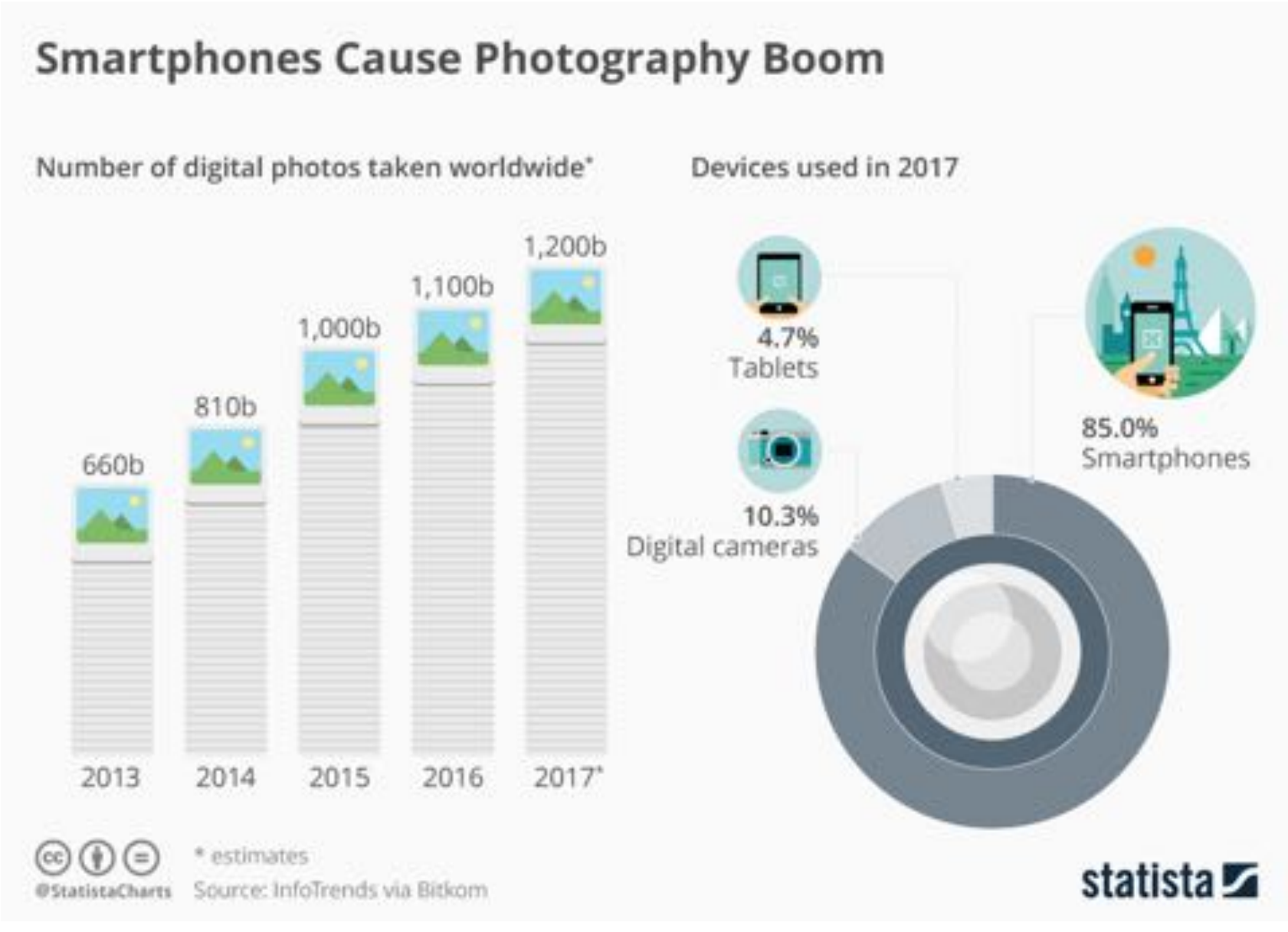
Tufte, Edward R. (1983). The Visual Display of Quantitative Information.

NIGEL HOLMES'S DESIGN PRINCIPLE

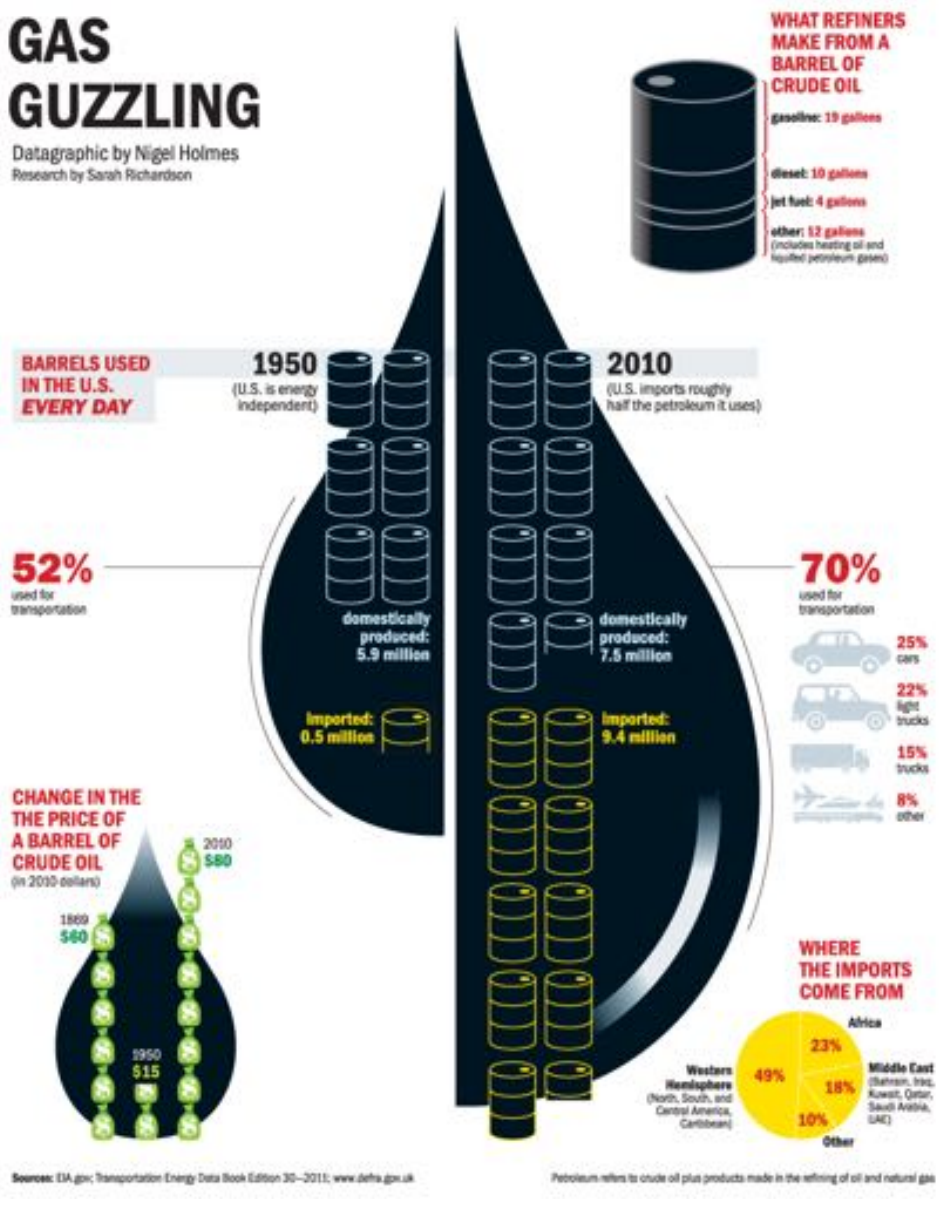
Use humor to instill affection in readers for numbers and charts



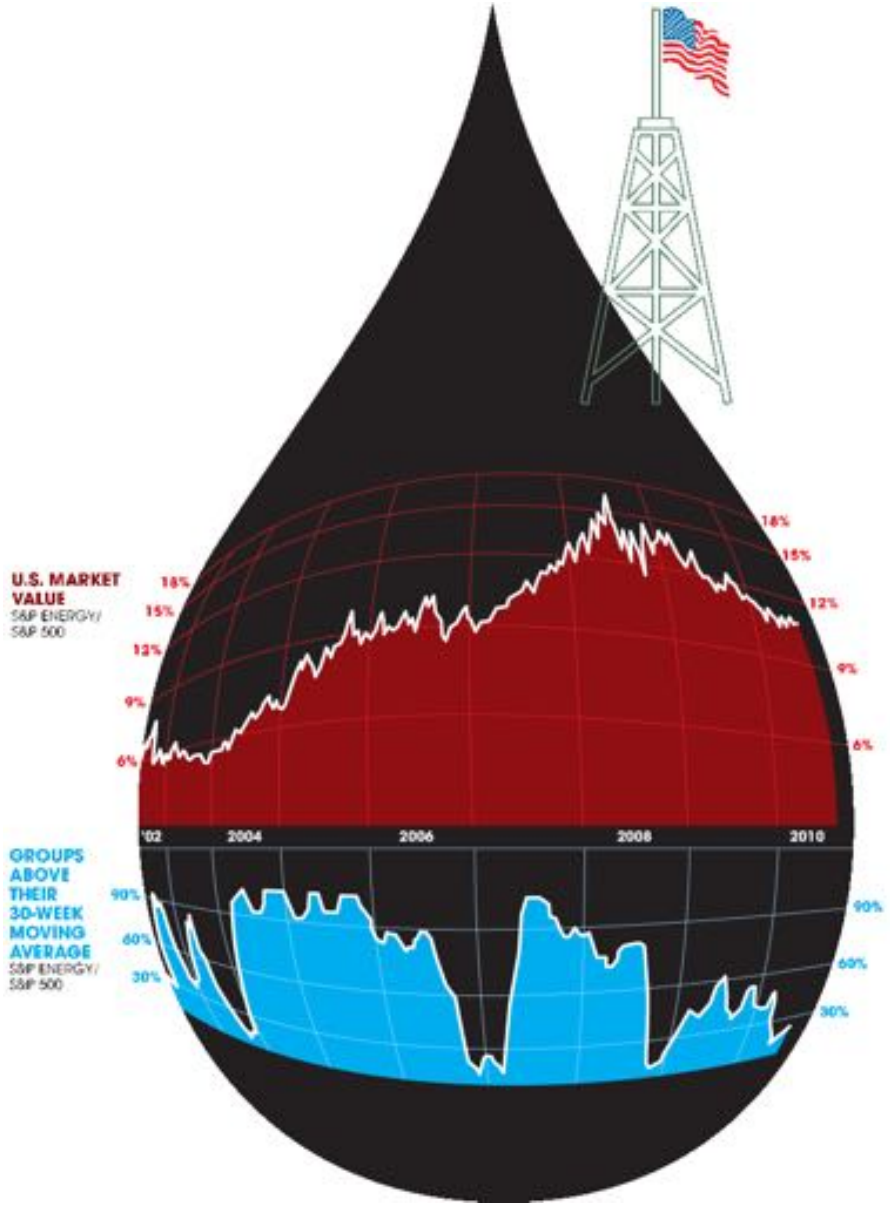
Nigel Holmes Website



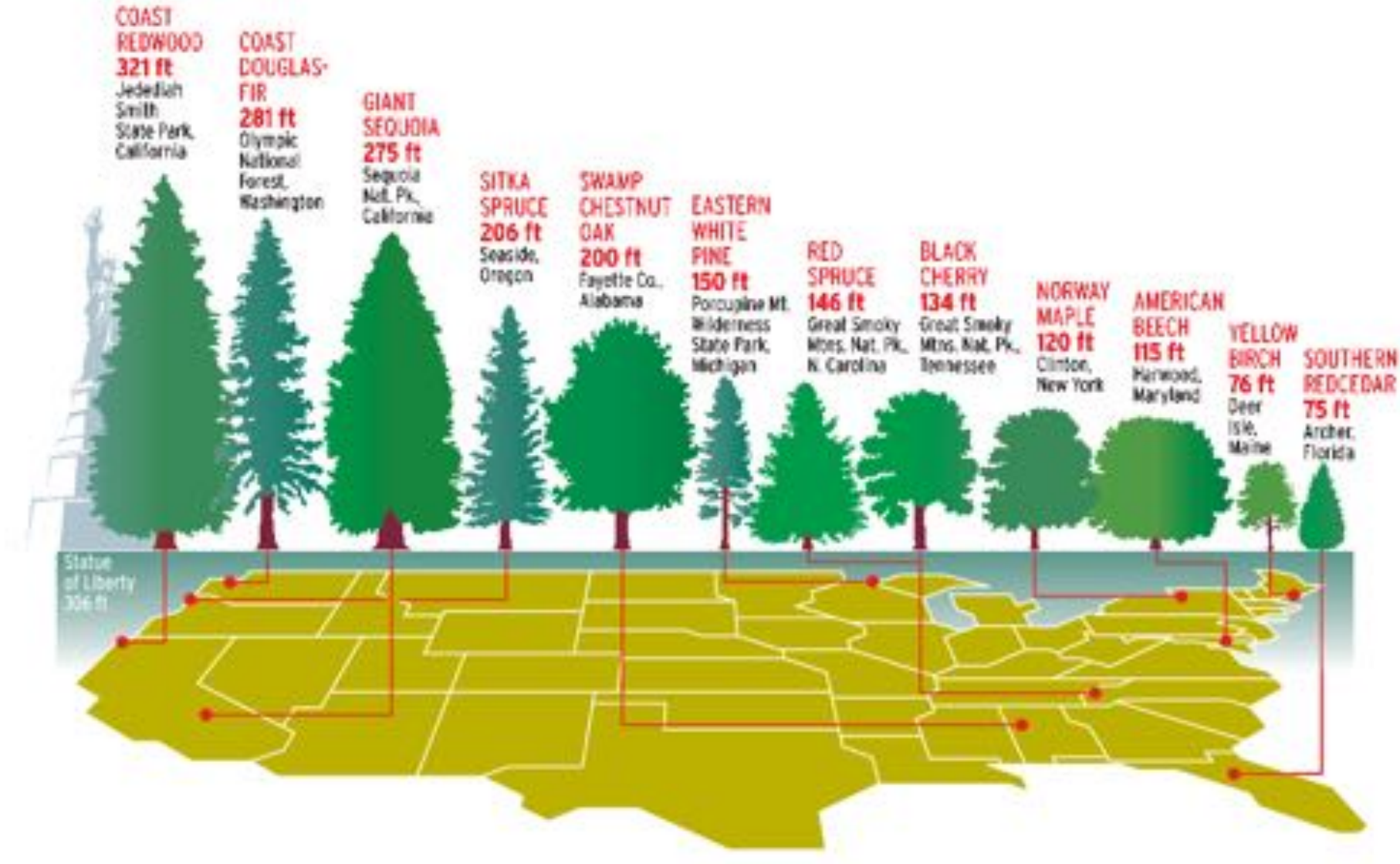
Statista - Smartphones Cause Photography Boom



Nigel Holmes Website

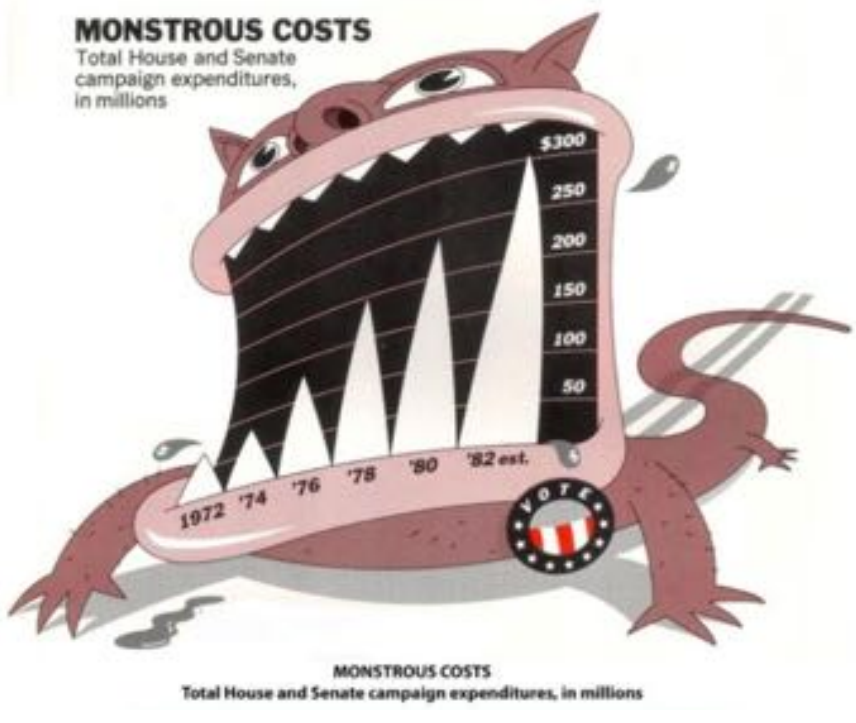


Nigel Holmes Website

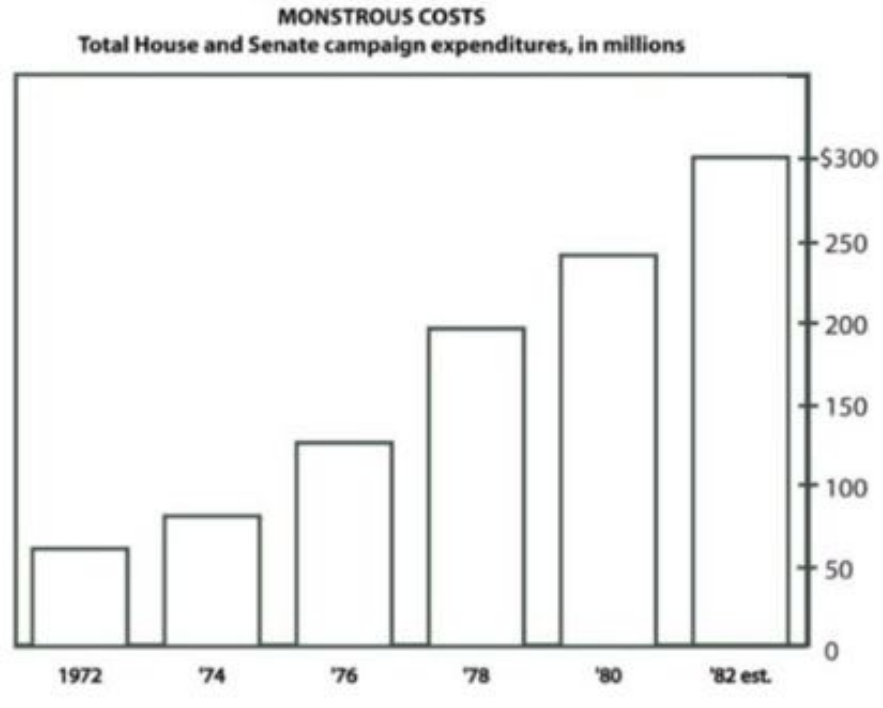

COAST REDWOOD
321 ft
Jedediah Smith State Park, California
COAST DOUGLAS-FIR
281 ft
Olympic National Forest, Washington
GIANT SEQUOIA
275 ft
Sequoia Nat. Pk., California
SITKA SPRUCE
206 ft
Seaside, Oregon
SWAMP CHESTNUT OAK
200 ft
Fayette Co., Alabama
EASTERN WHITE PINE
150 ft
Porcupine Mt. Wilderness State Park, Michigan
RED SPRUCE
146 ft
Great Smoky Mtns. Nat. Pk., N. Carolina
BLACK CHERRY
134 ft
Great Smoky Mtns. Nat. Pk., Tennessee
NORWAY MAPLE
120 ft
Clinton, New York
AMERICAN BEECH
115 ft
Harwood, Maryland
YELLOW BIRCH
76 ft
Deer Isle, Maine
SOUTHERN REDCEDAR
75 ft
Archer, Florida
Statue of Liberty
306 ft

USEFUL JUNK? [BATEMAN 2010]

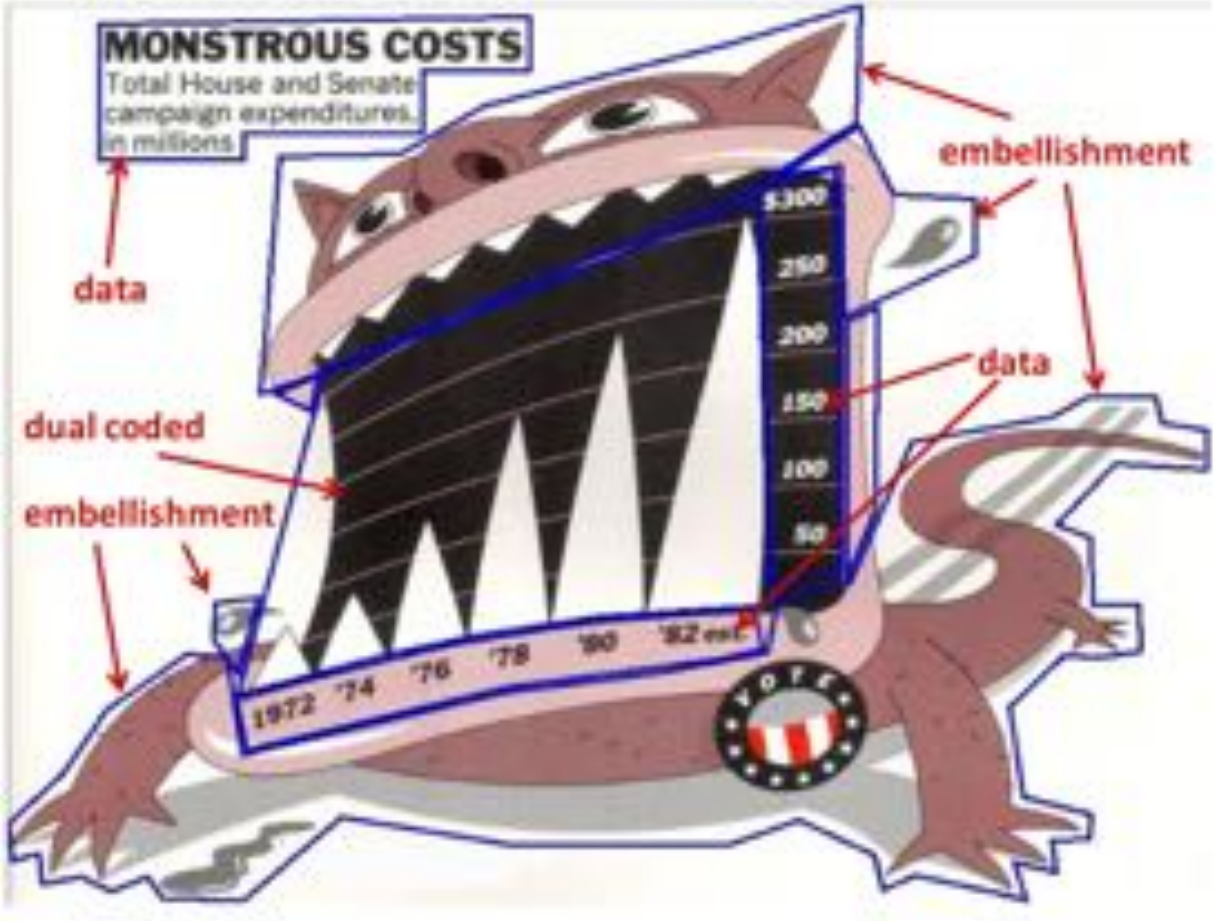
HOLMES



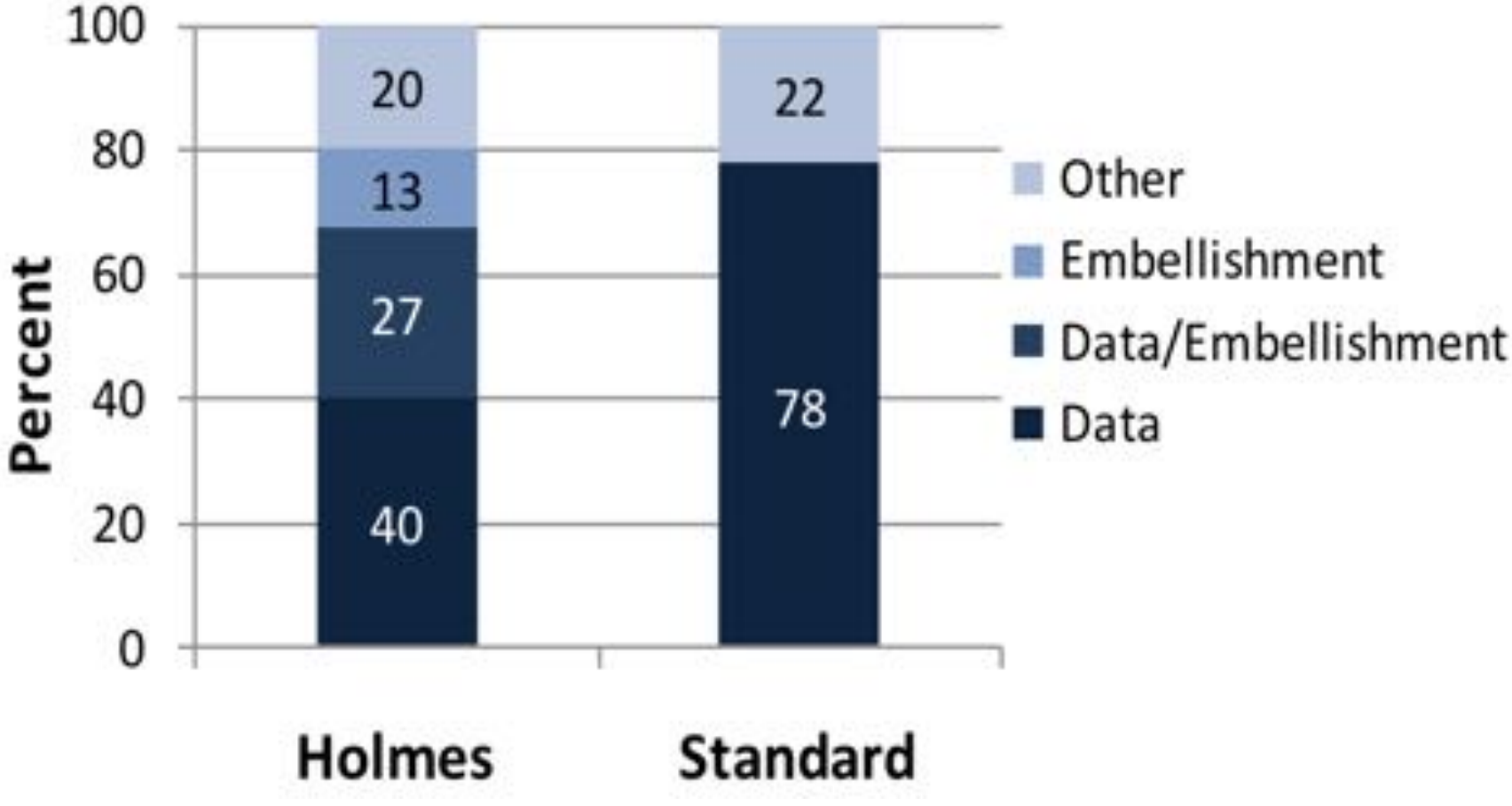
STANDARD



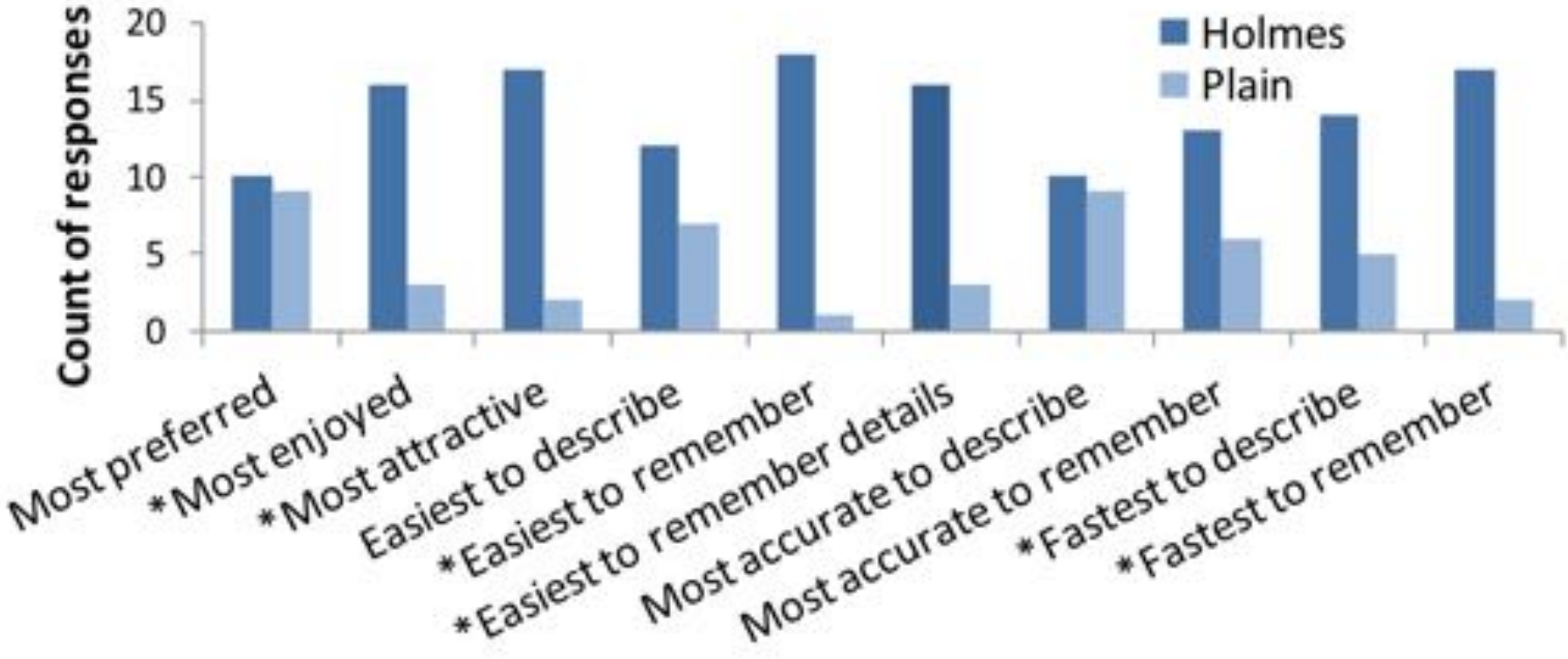
Bateman et al. Useful Junk? The Effects of Visual Embellishment on Comprehension and Memorability of Charts. ACM Conference on Human Factors in Computing Systems, Atlanta, GA, USA. 2010.



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

DATA JOIN WITH EMPTY SELECTION

D3 code

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

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(d => d);
```

Initial page

[Blank Page]

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

Final page

DATA JOIN WITH EMPTY SELECTION

D3 code

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

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(d => d);
```

Initial page

[Blank Page]

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

Final page

5

10

15

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

DATA JOIN WITH EMPTY SELECTION & PARENT NOT SELECTED

Javascript

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

d3.selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(d => d);
```

Initial page

[Blank Page]

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

Final page

DATA JOIN WITH EMPTY SELECTION & PARENT NOT SELECTED

Javascript

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

d3.selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(d => d);
```

Initial page

[Blank Page]

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

Final page

[Blank Page]

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

DATA JOIN WITH NON EMPTY SELECTION

Javascript

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

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(d => d);
```

Initial page

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

Final page

A

B

DATA JOIN WITH NON EMPTY SELECTION

Javascript

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

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(d => d);
```

Initial page

A

B

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

Final page

A

B

15

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

DATA JOIN WITH NON EMPTY SELECTION & PARENT NOT SELECTED

Javascript

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

d3.select('body');
d3.selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(d => d);
```

Initial page

A

B

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

Final page

DATA JOIN WITH NON EMPTY SELECTION & PARENT NOT SELECTED

Javascript

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

d3.select('body');
d3.selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(d => d);
```

Initial page

A

B

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

Final page

A

B

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

DATA JOIN WITH DIFFERENT ELEMENTS

Javascript

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

d3.select('body')
  .selectAll('div')
  .data(dataset)
  .enter()
  .append('span')
  .text(d => d);
```

Initial page

A

B

Final page

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

DATA JOIN WITH DIFFERENT ELEMENTS

Javascript

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

d3.select('body')
  .selectAll('div')
  .data(dataset)
  .enter()
  .append('span')
  .text(d => d);
```

Initial page

A

B

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

Final page

A

B

5 10 15

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

OUTLINE

- Function and esthetics, minimalistic visualizations
- D3 data join basics
  - Stress test
  - Customizing elements
  - Creating multiple elements per data point
- Loading data in D3

CUSTOMIZING HTML AND SVG ELEMENTS

```
//WITH HTML ELEMENTS
selection.attr(name[, value]) //set attributes, e.g., class
selection.style(name[, value[, priority]]) //set style properties
selection.text([value]) //set text content

//WITH SVG ELEMENTS
selection.attr(name[, value]) //place and size, e.g., x, width
selection.style(name[, value[, priority]]) //configure appearance
```

References

- selection.attr
- selection.style
- selection.text

SELECTING AND CONFIGURING THE PARENT

SELECTING

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

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('#svg')
  .attr('width', '300')
  .attr('height', '100');
```

```
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
  - Creating multiple elements per data point
- Loading data in D3

2 ELEMENTS PER DATA POINT WITH SVG

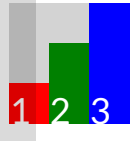
```
<svg width="60px" height="60px" style="background-color: lightgrey">

  <!-- 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"></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"></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"></text>

</svg>
```



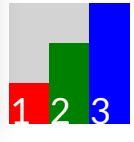
2 ELEMENTS PER DATA POINT WITH D3 (1)

```
<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', (d, i) => { return i * d.width; })
      .attr('y', d => { return 60 - d.height; })
      .attr('width', d => d.width)
      .attr('height', d => d.height)
      .attr('fill', d => d.color);

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



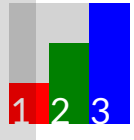
2 ELEMENTS PER DATA POINT WITH D3 (2)

```
<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 enterSelection = d3.select('#char21')
      .selectAll('rect')
      .data(dataset)
      .enter();

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

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



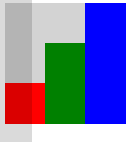
2 ELEMENTS PER DATA POINT WITH D3

```
<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', (d, i) => { return i * d.width; })
      .attr('y', d => { return 60 - d.height; })
      .attr('width', d => d.width)
      .attr('height', d => d.height)
      .attr('fill', d => d.color)

      .append('text')
      .attr('x', (d, i) => { return i * d.width; })
      .attr('y', d => 60)
      .attr('font-size', '18px')
      .attr('fill', "white")
      .text(d => 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 CHARTS

You can load data in 2 ways:

1. Hard-code the data in the page
2. Loading the data from a remote URL*

D3 promises** to make HTTP requests for data:

1. d3.csv(): request CSV files (not typed)
2. d3.json(): request JSON files (typed)

* For security reasons browsers will not access or load local data files

** Promises are JavaScript interfaces for making asynchronous operations. Other node libraries (e.g., fetch, axios) that are designed to make HTTP requests can also be used to fetch data.

ABOUT PROMISES

```
let myPromise = new Promise((resolve, reject) => {
  // resolve(...) is called if successful
  // reject(...) is called if fail

  resolve([5, 10, 15]);
});

myPromise.then((result) => {
  console.log('result is:', result);
});
```

LOADING CSV FILES WITH D3.CSV (1)

Create the data file. First row used for column names, no spaces after the comma!

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

Loading the data

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

Print-out of the data once loaded

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

LOADING CSV FILES WITH D3.CSV (2)

Loading the data and setting variable types

```
d3.csv("cars.csv", function(d) {
  return {
    year: new Date(+d.Year, 0, 1), // convert "Year" column to Date
    make: d.Make,
    model: d.Model,
    length: +d.Length // convert "Length" column to number
  };
}, function(error, rows) {
  console.log(rows);
});
```

Print-out of the data once loaded

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

See d3-dsv documentation

CONVERTING STRINGS TO NUMBERS

With `parseInt()` and `parseFloat()`

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

With unary + operator

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

Coersion with the unary + operator is preferred because it is faster and shorter to write.

LOADING JSON FILES WITH D3.JSON

JSON stands for Javascript object notation

Create the data file. Key value-pairs as strings.

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

Loading the data

```
d3.json("cars.json").then(data => console.log(data));
```

Print-out of the data once loaded

```
[{year: 1997, make: "ford", model: "e350", length: 2.34},
 {year: 2000, make: "mercury", model: "cougar", length: 2.38}]
```

See the d3.json documentation

What will appear on the page?

data.csv

```
Color
Red
Green
Blue
```

```
<p>Orange</p>
<script>
  d3.csv('data.csv').then(data => {
    d3.select('body')
      .selectAll('p')
      .data(data)
      .enter()
      .append('p')
      .text(d => 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?

data.csv

```
Color
Red
Green
Blue
```

```
<p>Orange</p>
<script>
  d3.csv('data.csv').then(data => {
    d3.select('body')
      .selectAll('p')
      .data(data)
      .enter()
      .append('p')
      .text(d => 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

Solution: B