CS 499/549: Visual Analytics

Visualization Programming (2)

JavaScript Frameworks and Svelte

Minsuk Kahng
Assistant Professor
School of Electrical Engineering and Computer Science
Oregon State University

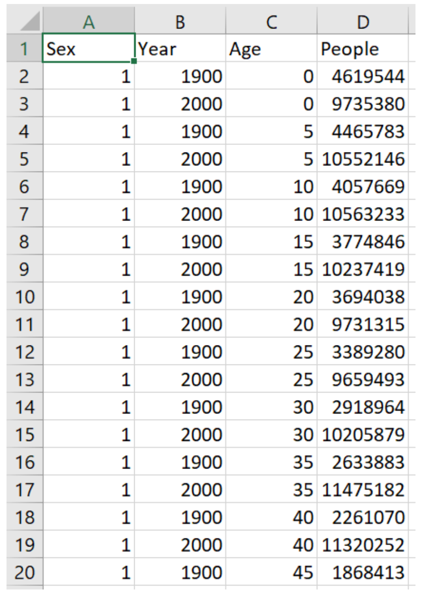
Today's topics:

1. Visualization Libraries
    - Vega-Lite
2. SVG with HTML/CSS
    - Draw a bar chart, scatterplot, and line chart
3. In-Class Activity
4. JavaScript Warm-up

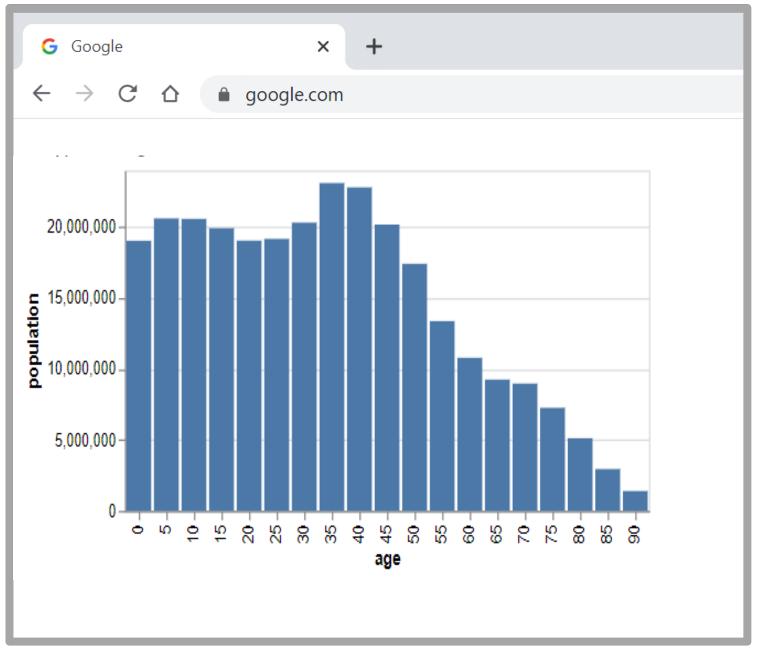
Review: HTML/CSS and SVG

Visualization Programming

- Methods to systematically draw SVG elements from data
  - For each data item, create a mark (e.g., <rect>)
  - For each data attribute, style with an appropriate channel (e.g., color)
- Let's dive into the low-level approach.



	A	B	C	D
1	Sex	Year	Age	People
2	1	1900	0	4619544
3	1	2000	0	9735380
4	1	1900	5	4465783
5	1	2000	5	10552146
6	1	1900	10	4057669
7	1	2000	10	10563233
8	1	1900	15	3774846
9	1	2000	15	10237419
10	1	1900	20	3694038
11	1	2000	20	9731315
12	1	1900	25	3389280
13	1	2000	25	9659493
14	1	1900	30	2918964
15	1	2000	30	10205879
16	1	1900	35	2633883
17	1	2000	35	11475182
18	1	1900	40	2261070
19	1	2000	40	11320252
20	1	1900	45	1868413



Scalable Vector Graphics (SVG)

- A markup language for describing two-dimensional based vector graphics
- Elements
  - <rect>: Rectangle. Specified by x, y, width, and height.
  - <circle>: Circle or point. Specified by cx, cy, and r
  - <line>: Line. Specified by x1, x2, y1, and y2
  - <text>: Text
  - <path>: Line segments?

Scalable Vector Graphics (SVG)

- Style attributes
  - fill: color of element
  - stroke: color of border
  - stroke-width: size of border
- Example:
  - `<rect id=“my-rectangle” x=“10” y=“10” width=“80” height=“15” />`
  -
    ```
    #my-rectangle {
        fill: green;
        stroke: #0000ff; // blue
        stroke-width: 3;
    }
    ```

Tips

- Strongly recommended to use "id" or "class" tags, rather than using style tags directly on DOM
- Use consistent indents for readability
- Using <g> can make your code more organized
- "google“ for reference. Mozilla pages are useful.

JavaScript Frameworks

Goal

- Last time, we manually write svg elements (e.g., rect) for every data item.
- Goal: We want to programmatically create a visualization from data.
  - For loop to draw a <rect> for each data item
  - Calculate "height" from data attributes
- JavaScript will let you do this.

JavaScript frameworks

- It’s possible to do it with vanilla JavaScript without any libraries or framework.
- But, with frameworks, it’s much easier to build highly dynamic, interactive applications.
  - React, Vue.js, Angular, Svelte

JavaScript frameworks

No framework

```
<script>
const tasks = [
  {name: “shopping”}, {name: “call Fred”}]

function buildTodoList() {
  const element =
    document.getElementById(“todo”);
  tasks.forEach(task => {
    const item =
      document.createElement('li');
    const span =
      document.createElement('span');
    const textContent =
      document.createTextNode(task.name);
    span.appendChild(textContent);
    element.appendChild(item);
  });
}
buildTodoList();
</script>


<ul id=“todo”></ul>
```

With framework

```
<script>
const tasks = [
  {name: “shopping”}, {name: “call Fred”}]
</script>

{#each tasks as task}
  <ul id=“todo”>
    <li><span>{task.name}</span></li>
  </ul>
{/each}
```

JavaScript frameworks: Insert?

No framework

```
<script>
const tasks = [
  {name: “shopping”}, {name: “call Fred”}];

const button = document.getElementById(“insert-button”);
button.onclick = (element) => {
  const taskName = element.getAttribute(“value”);
  const element =
    document.getElementById(“todo”);
  const item =
    document.createElement('li');
  const span =
    document.createElement('span');
  const textContent =
    document.createTextNode(task.name);
  span.appendChild(textContent);
  element.appendChild(item);
};
</script>


<ul id=“todo”></ul>

<button id=“insert-button” value=“shopping”>
  New task: shopping
</button>
```

With framework

```
<script>
const tasks = [
  {name: “shopping”}, {name: “call Fred”}];

function insert(taskName) {
  tasks.push({name: taskName});
}
</script>

{#each tasks as task}
  <ul id=“todo”>
    <li><span>{task.name}</span></li>
  </ul>
{/each}

<button on:click={() => insert(“shopping”)>
  New task: shopping
</button>
```

JavaScript frameworks: Insert?

- Key idea: “reactive”
- Data is served as a “state” of a program.
- Developers just need to specify how state will be displayed in HTML.

```
<script>
const tasks = [
  {name: “shopping”}, {name: “call Fred”}];

function insert(taskName) {
  tasks.push({name: taskName});
}
</script>

{#each tasks as task}
  <ul id=“todo”>
    <li><span>{task.name}</span></li>
  </ul>
{/each}

<button on:click={() => insert(“shopping”)>
  New task: shopping
</button>
```

In-Class Activity

Creating visualizations using JavaScript frameworks