Run `git pull` in the main branch to follow along today.

D3.js (Part 2)

DSC 106: Data Visualization

Sam Lau

UC San Diego

Announcements

Project 2 peer grading due on Tuesday.

Project 3 checkpoint due on Tuesday.

FAQs:

1. I'm getting 404 errors when working on the lab! Check that you're using relative URLs and they match your folder structure.

Project 2 Peer Feedback

Opportunity to get feedback from your peers.

"I like / I wish / What if?" format.

Worth 5% of your final grade, graded by completion.

Project 3: Interactive Visualization

Choose a health dataset (can reuse Project 2 data).

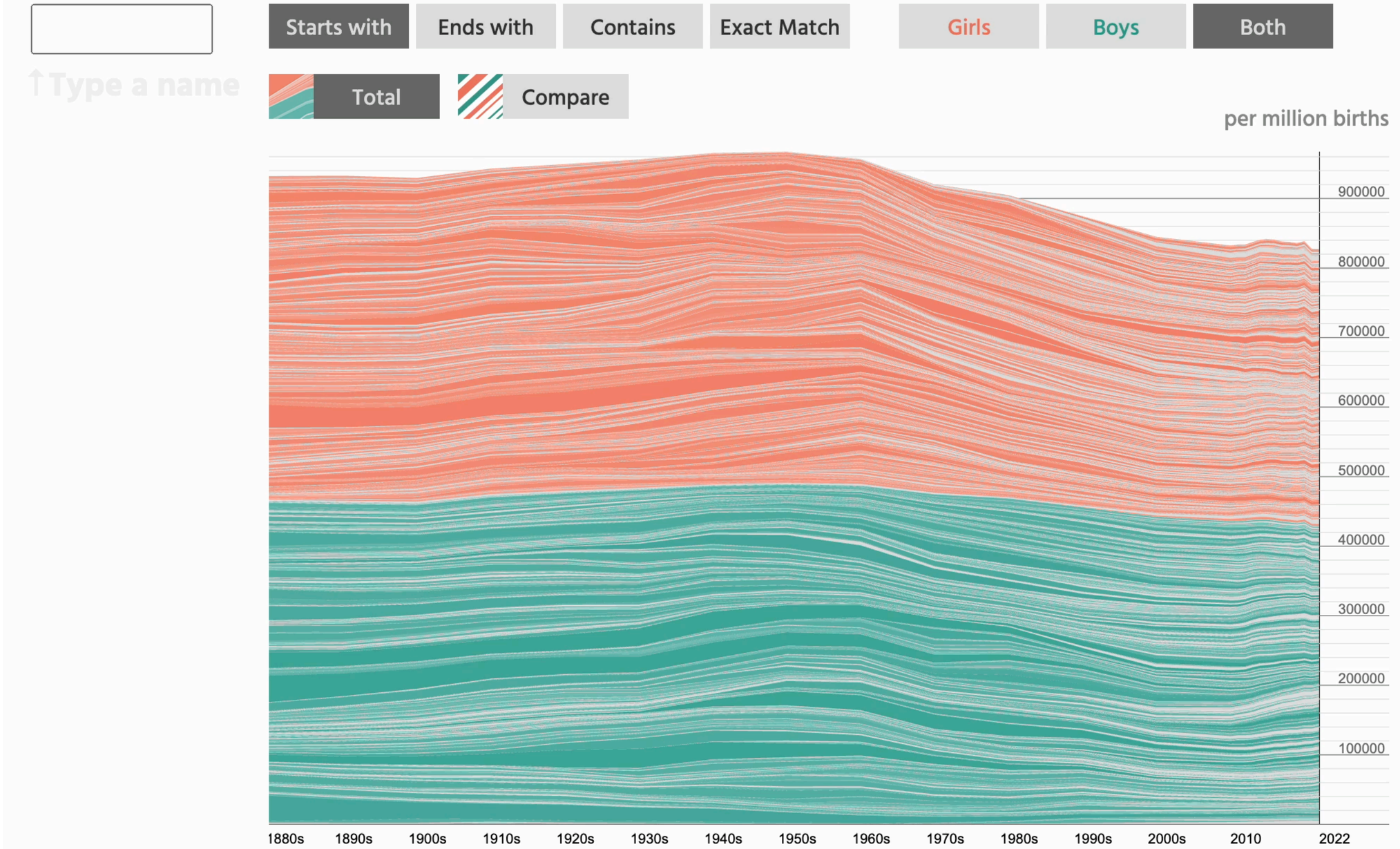
Create **one** interactive graphic to let readers explore the data.

E.g. panning, zooming, brushing, annotations, etc.

Must use D3.

Must complete in teams of 3-4.

Pro-tip: Explore lots of options using pen-and-paper. Then, keep scope of project very tight! Do one thing well.



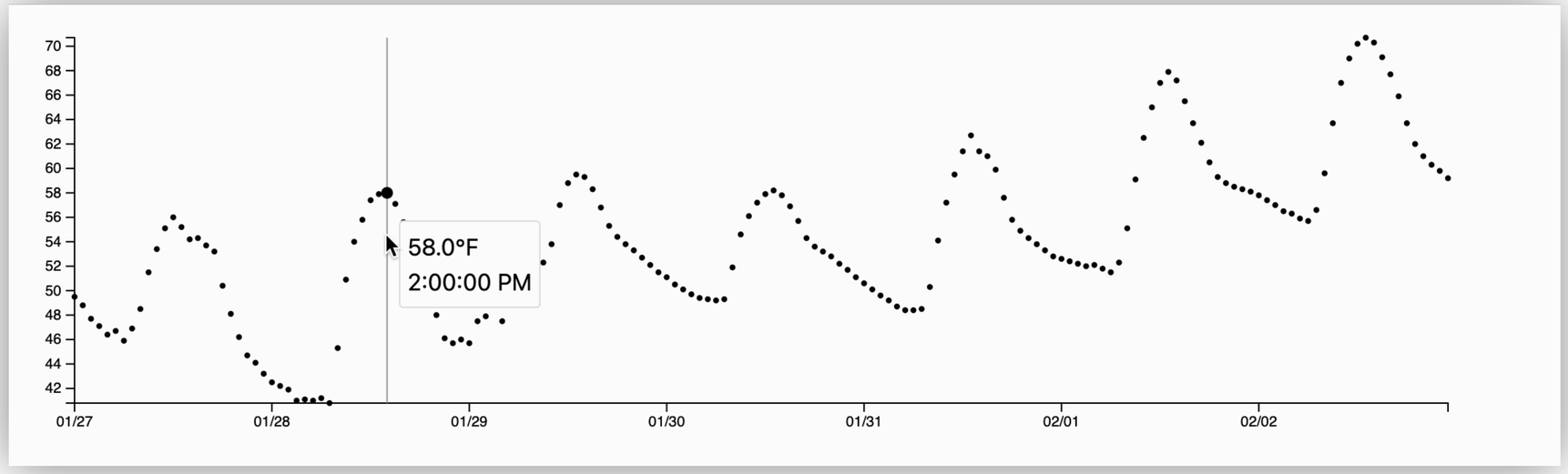
https://namerology.com/baby-name-grapher/

D3

Today: Making an interactive scatterplot

Before:

After:



Step 1: Using D3 instead of plain JS

Before:

After:

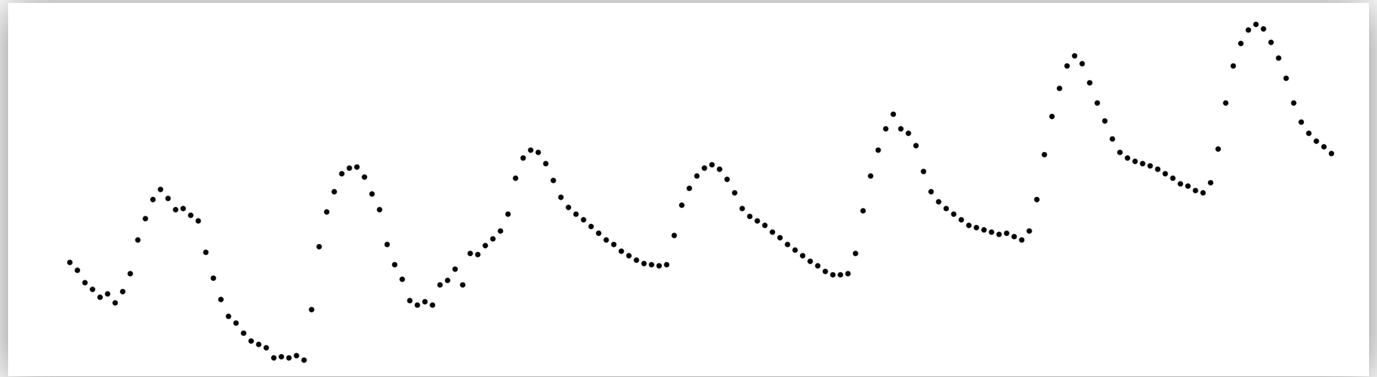
But in D3!

Demo: `d3-lecture/weather01`

Step 2: Making circles and using d3 scales

Before:

After:



Demo: `d3-lecture/weather02`

Making circles

Before:

```
svg
  .selectAll('text')
  .data(weatherData.hourly.temperature_2m)
  .join('text')
  .attr('x', (d, i) => i * 5)
  .attr('y', (d) => 500 - d * 6)
  .text((d) => d);
```

After:

```
svg
  .selectAll('circle')
  .data(weatherData.hourly.temperature_2m)
  .join('circle')
  .attr('cx', (d, i) => xScale(i))
  .attr('cy', (d) => yScale(d))
  .attr('r', 2);
```

Just needed to swap out text with circle + set the right attributes.

Scales

Before:

```
.attr('cx', (d, i) => i * 5)
.attr('cy', (d) => 500 - d * 6)
```

Magic numbers!

After:

```
.attr('cx', (d, i) => xScale(i))
.attr('cy', (d) => yScale(d))
```

D3 scales

```
const xScale = d3
  .scaleLinear()
  .domain([0, weatherData.hourly.temperature_2m.length - 1])
  .range([margin.left, width - margin.right]);
```

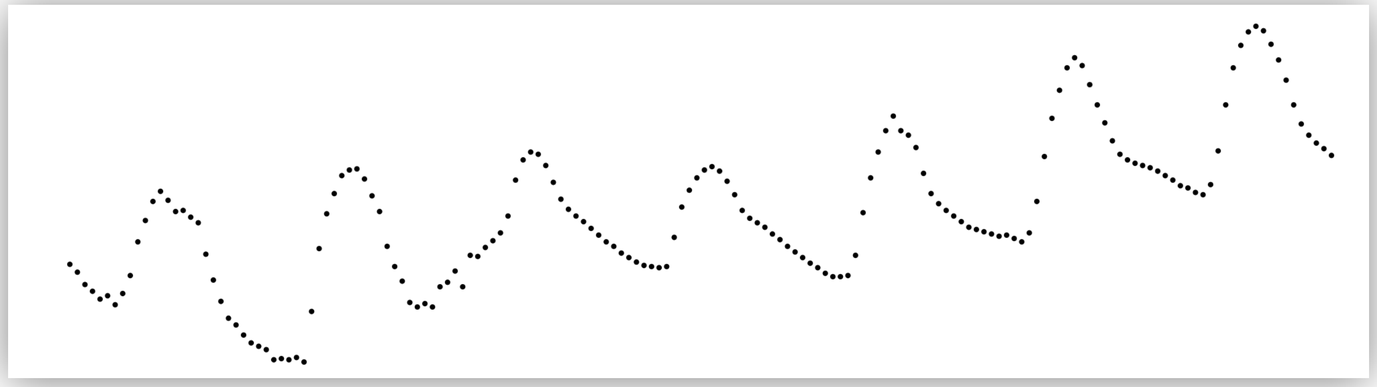
Domain = possible inputs

Range = possible outputs

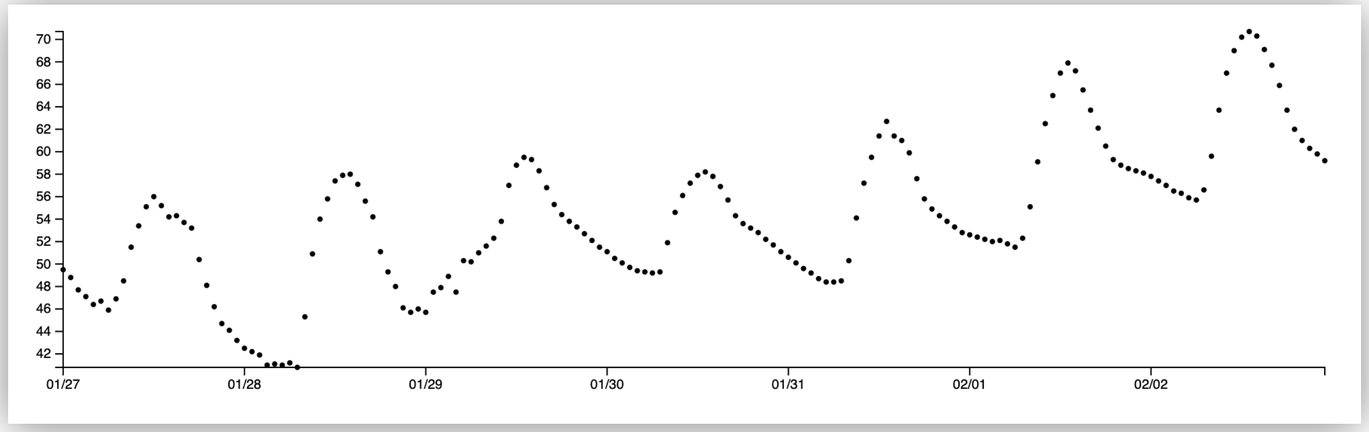
D3 scales will automatically make plot fit the space.

Step 3: Adding axes

Before:



After:



Demo: `d3-lecture/weather03`

Axes

```
const yAxis = d3.axisLeft(yScale);
```

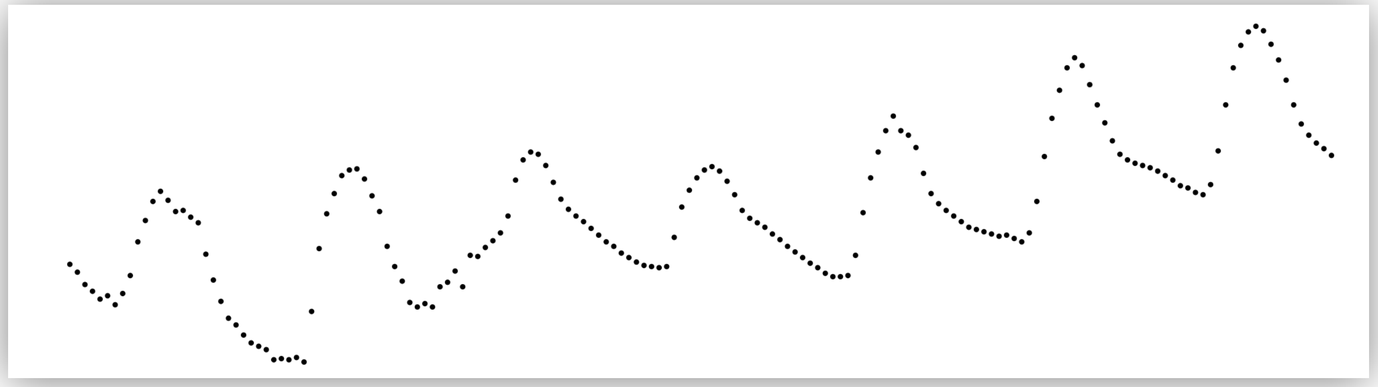
Creates a D3 axis object

```
svg
  .append('g')
  .attr('class', 'y axis')
  .attr('transform', `translate(${margin.left}, 0)`)
  .call(yAxis);
```

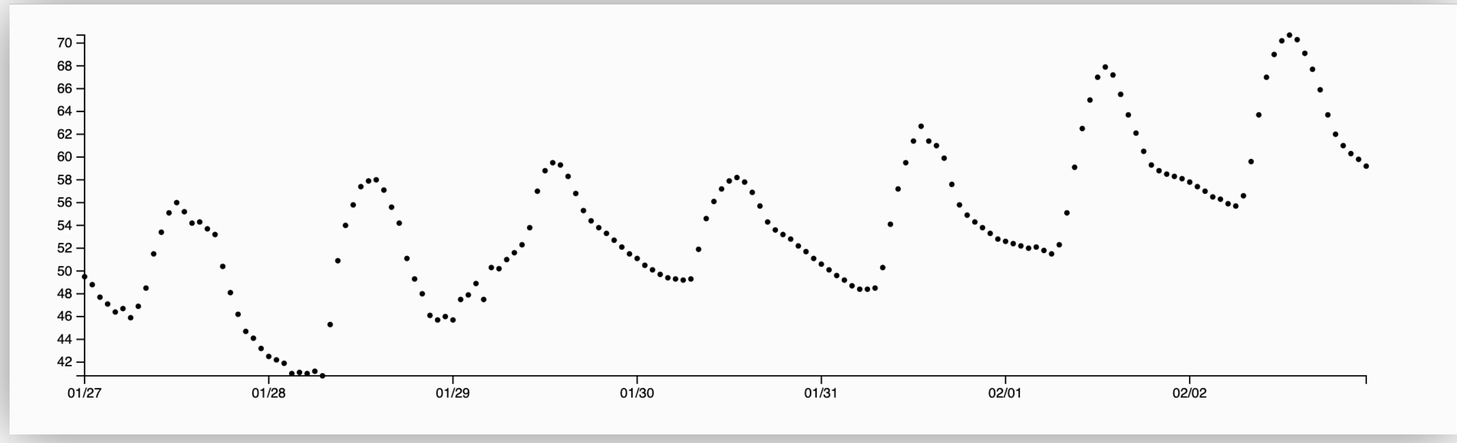
Creates an SVG <g> object, then draws axis into it

Step 4: Adding a basic tooltip

Before:



After:



Demo: `d3-lecture/weather04`

Making a tooltip

```
const tooltip = d3
  .select('body')
  .append('div')
  .attr('class', 'tooltip')
  .style('position', 'absolute')
  .style('visibility', 'hidden')
  .style('background-color', 'white')
  .style('border', '1px solid #ddd')
  .style('padding', '5px')
  .style('border-radius', '3px');
```

Creates a , styles it, and hides it so that it'll only show up with interaction

Adding interaction

```
.on('mouseover', function (event, d) {
  d3.select(this).attr('r', 4); // Increase circle size on hover


  tooltip.style('visibility', 'visible').text(`${d.toFixed(1)}°F`);
})
```

D3 version of event listener + handler

Adding interaction

```
.on('mouseover', function (event, d) {
  // Increase circle size on hover

  tooltip.style('visibility', 'visible').text(`${d.toFixed(1)}°F`);
})
```

When a circle is moused over...

D3 version of event listener + handler

Adding interaction

```
.on('mouseover', function (event, d) {
  d3.select(this).attr('r', 4); // Increase circle size on hover
                                                    ${d.toFixed(1)}°F`);
})
```

Make the circle’s radius larger

D3 version of event listener + handler

Adding interaction

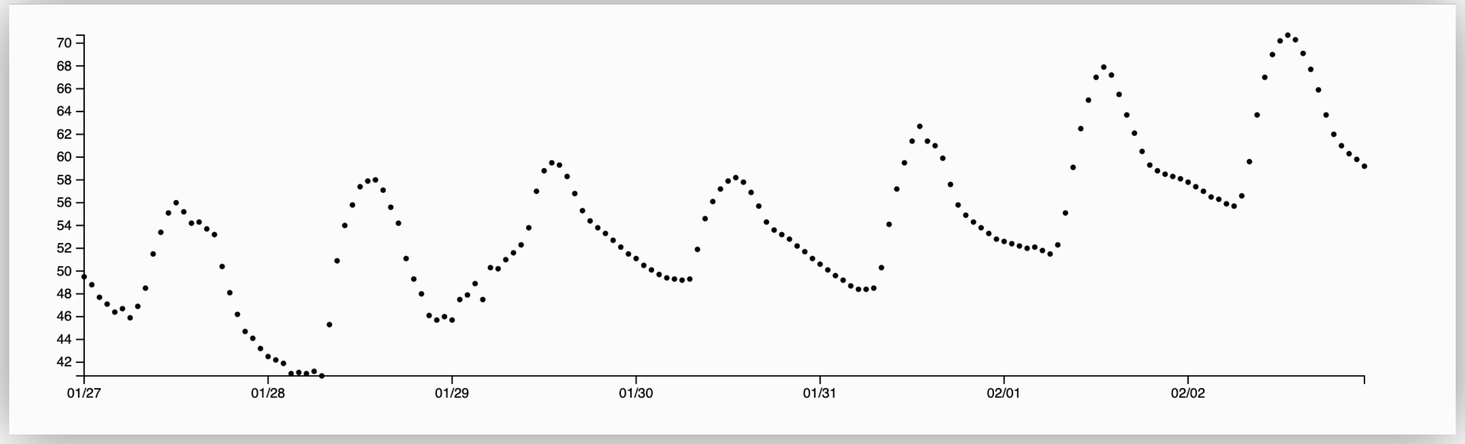
```
.on('mouseover', function (event, d) {
  d3.select(this).attr('r', 4); // Increase circle size on hover

  tooltip.style('visibility', 'visible').text(`${d.toFixed(1)}°F`);
})
```

Make tooltip visible and set its text

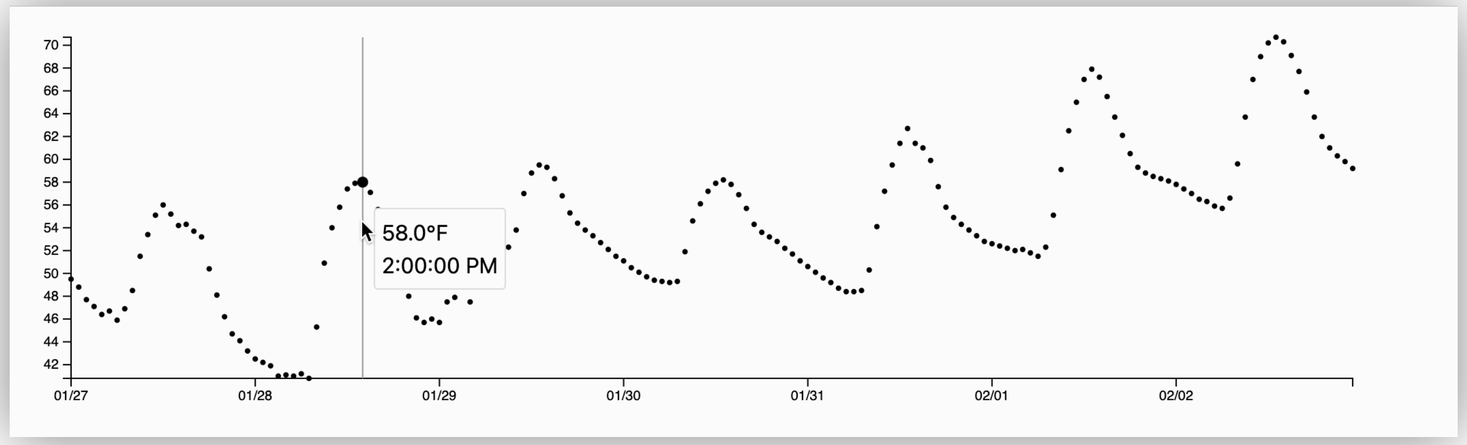
D3 version of event listener + handler

Step 5: Improving our tooltip

Before:



After:



Demo: `d3-lecture/weather05`

Interacting with the plot, not just points

```
// Create a rect overlay for mouse tracking
const overlay = svg
  .append('rect')
  .attr('class', 'overlay')
  .attr('x', margin.left)
  .attr('y', margin.top)
  .attr('width', width - margin.left - margin.right)
  .attr('height', height - margin.top - margin.bottom)
  .style('fill', 'none')
  .style('pointer-events', 'all');
```

Interaction trick:
Add an invisible rectangle just to capture mouse events

Listening for mouse events on the parent <svg> tag also ok

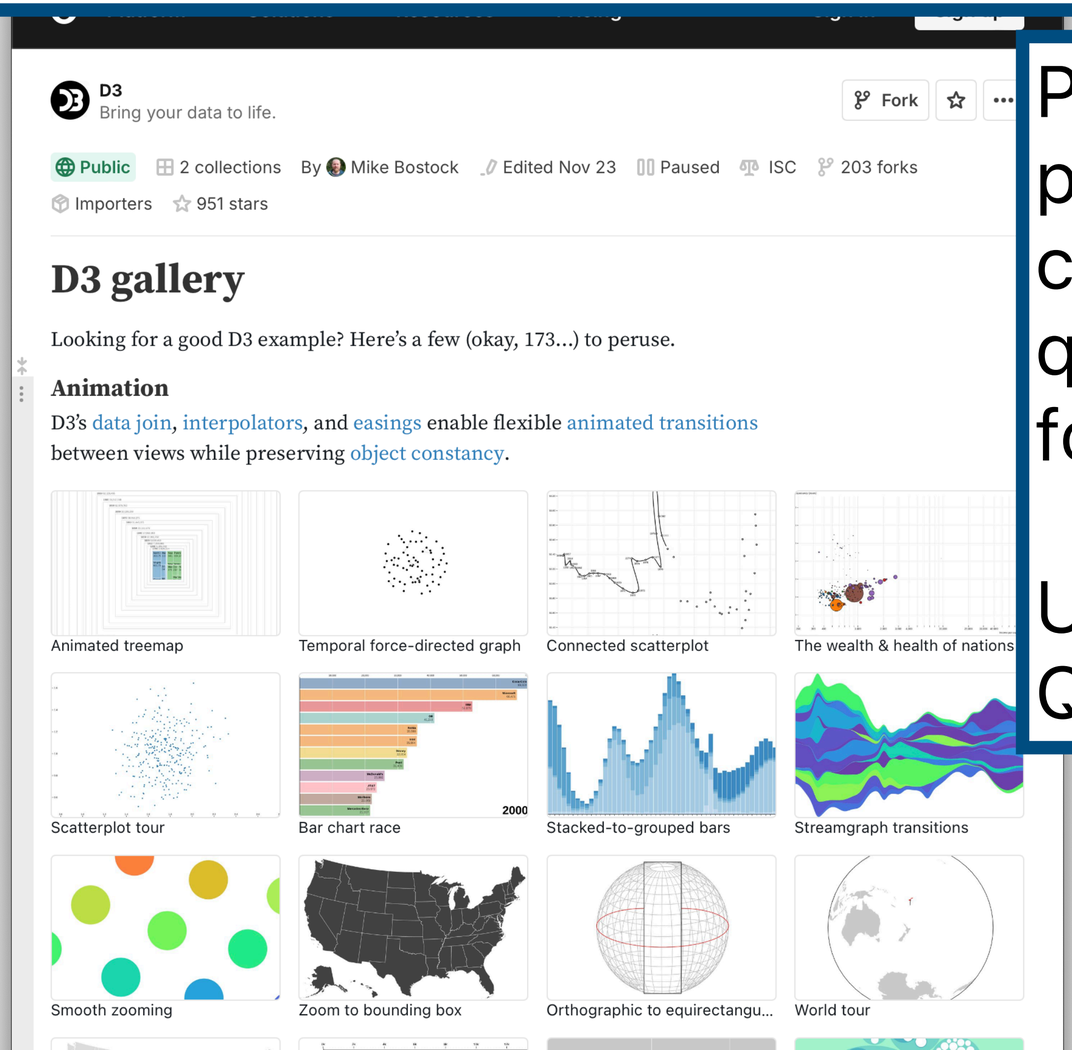
Improving interaction

```
.on('mousemove', function (event) {
  const mouseX = d3.pointer(event)[0];
  const xDate = xScale.invert(mouseX);

  // Find the closest data point
  const bisect = d3.bisector((d) => new Date(d)).left;
  const index = bisect(weatherData.hourly.time, xDate);
  const temp = weatherData.hourly.temperature_2m[index];
  const time = new Date(weatherData.hourly.time[index]);
```

Challenge: since we're not hovering directly over points, we have to use the mouse position to find nearest point

You Try: Explain D3 code

https://observablehq.com/@d3/gallery



Pick a simple visualization (scatter plot, line plot, bar chart). Explain the code to your neighbor, then write a question about the code using this format:

URL: …
Question: …

tryclassbuzz.com
Code: **explain-d3**