DDA 2003 / MDS 6112
Visual Analytics / Data Visualization

Lecture 8
D3

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Outline

- Data joins and basic interactivity
- Multiple views and advanced interactivity
- Advanced concepts

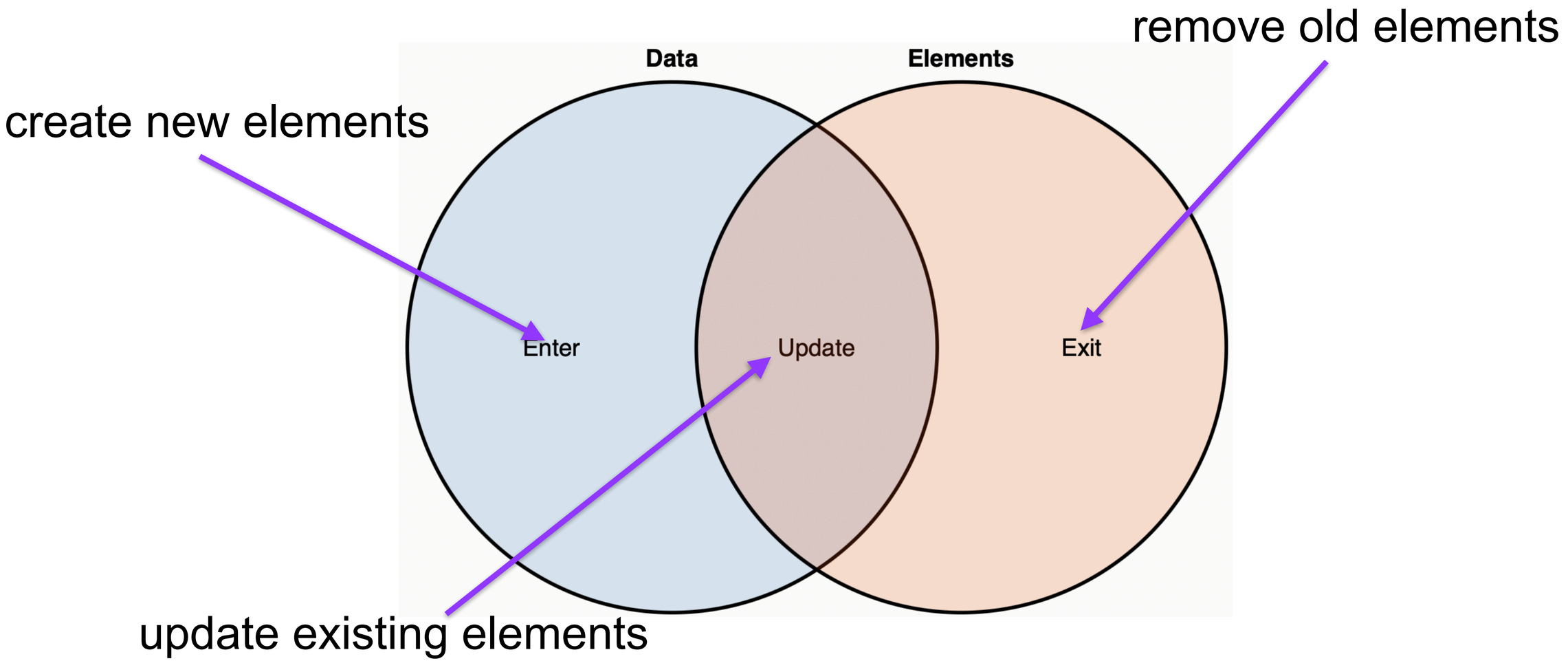
Outline

- Data joins and basic interactivity
- Multiple views and advanced interactivity
- Advanced concepts

Data joins and basic interactivity

- Instead of removing and redrawing visualizations each time new data arrives, we want to update only affected components to improve loading times and create smooth transitions
- Enter, update, and exit
  - Enter:What happens to new data values without existing, associated DOM elements?
  - Update: What happens to existing elements which have changed?
  - Exit: What happens to existing DOM elements which are not associated with data anymore?

Data joins and basic interactivity



Data joins and basic interactivity

- Join() shortcut for the enter-update-exit pattern
  - Alternatively use the join() method which is simpler and more convenient for many cases. Instead of specifying enter, update, and exit operations separately, join() handles all three stages automatically. New elements will be added, existing elements will be updated, and obsolete elements will be removed

```
function updateChart(data) {
  svg.selectAll('circle')
      .data(data)
    .join('circle')
      .attr('fill', '#707086')
      .attr('r', d => d)
      .attr('cx', (d,index) => (index * 80) + 50)
      .attr('cy', 80);
}
```

Data joins and basic interactivity

- Handle user input
  - Bind an event listener to any DOM element using d3.select().on()

Data joins and basic interactivity

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

```
<!-- ... -->
<body>
    <!-- HTML form -->
    <div>
        <label for="radius-slider">Radius: <span id="radius-value">60</span></label>
        <input type="range" min="1" value="60" max="80" id="radius-slider">
    </div>

    <!-- Empty SVG drawing area -->
    <svg id="chart" width="200" height="200"></svg>

    <script src="js/d3.v6.min.js"></script>
    <script src="js/main.js"></script>
</body>
</html>
```

```
const svg = d3.select('svg');

// Show circle with initial radius of 60px
const circle = svg.append('circle')
    .attr('cx', 100)
    .attr('cy', 100)
    .attr('fill', 'none')
    .attr('stroke', 'green')
    .attr('r', 60);

function updateCircle(radius) {
  circle.attr('r', radius);
}
```

```
d3.select('#radius-slider').on('input', function() {
  // Update visualization
  updateCircle(parseInt(this.value));

  // Update label
  d3.select('#radius-value').text(this.value);
});
```

Data joins and basic interactivity

- Interaction process
  - Add global event listeners (e.g., checkboxes, sliders, …)
  - Update configurations or data
  - Update the visualization accordingly
  - Add chart-specific events

```
svg.selectAll('circle')
      .data(data)
    .join('circle')
    .attr('fill', 'green')
    .attr('r', 4)
    .attr('cx', d => vis.xScale(d.x))
    .attr('cy', d => vis.yScale(d.y))
    .on('mouseover', d => console.log('debug, show tooltip, etc.'))
```

- Animated transitions
  - Apply transition() method that creates smooth, animated transitions between states
  - Default time span is 250 milliseconds
  - Use duration() to specify the value
  - Use delay() to delay a transition

```
d3.selectAll('circle')
   .transition()
   .duration(3000)
   .attr('fill', 'blue');
```

- Pros of animated transitions
  - Transitions show what is happening between states and add a sense of continuity to your visualization
  - Animations can draw the user's attention to specific elements or aspects
  - Animations can provide the user with interactive feedback

- Cons of animated transitions
  - Too many transitions will confuse the user (e.g., overused PowerPoint effects)
  - If the transition is not continuous, animations look strange and can even be deceiving based on the interpolation used
  - Animation across many states is the least effective use case for data analysis tasks. In this case, use a static comparison of several charts/images (e.g., small multiples) instead of creating video-like animations

Data joins and basic interactivity

- Tooltips
  - Reveal more details and information to the users

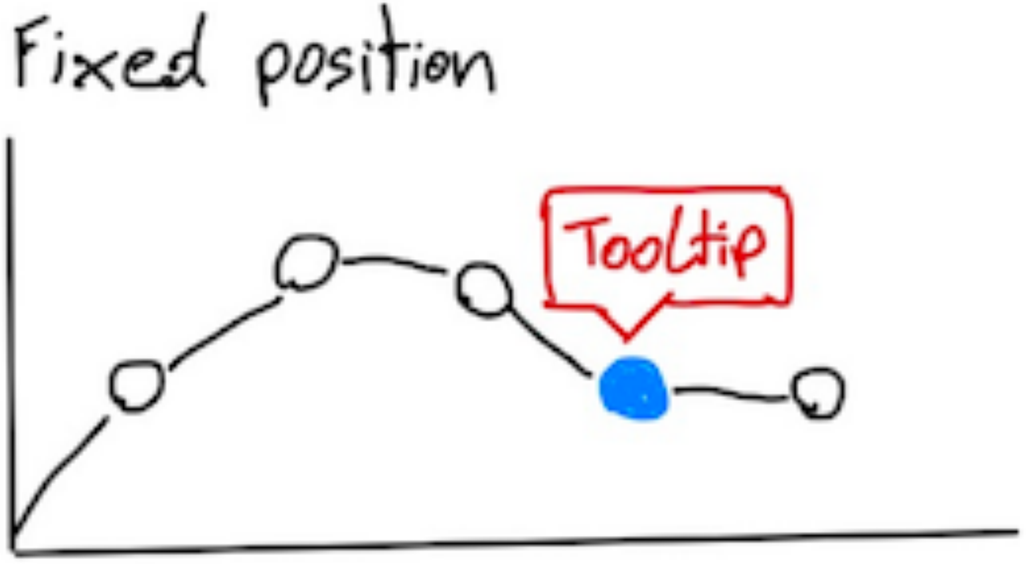
```
<div id="tooltip"></div>
```

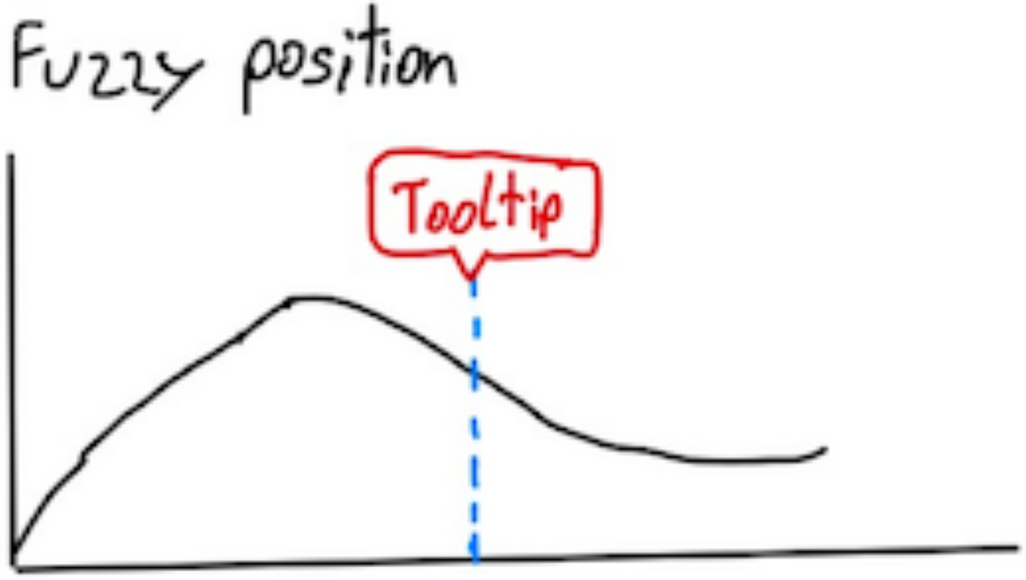
```
#tooltip {
  position: absolute;
  display: none;
  /* ... other tooltip styles ... */
}
```

```
myMarks
    .on('mouseover', (event,d) => {
      d3.select('#tooltip')
        .style('display', 'block')
        // Format number with million and thousand separator
        .html(`<div class="tooltip-label">Population</div>${d3.format(',')(d.population)}`);
    })
    .on('mousemove', (event) => {
      d3.select('#tooltip')
        .style('left', (event.pageX + vis.config.tooltipPadding) + 'px')
        .style('top', (event.pageY + vis.config.tooltipPadding) + 'px')
    })
    .on('mouseleave', () => {
      d3.select('#tooltip').style('display', 'none');
    });
```

Data joins and basic interactivity

- Tooltips for path elements





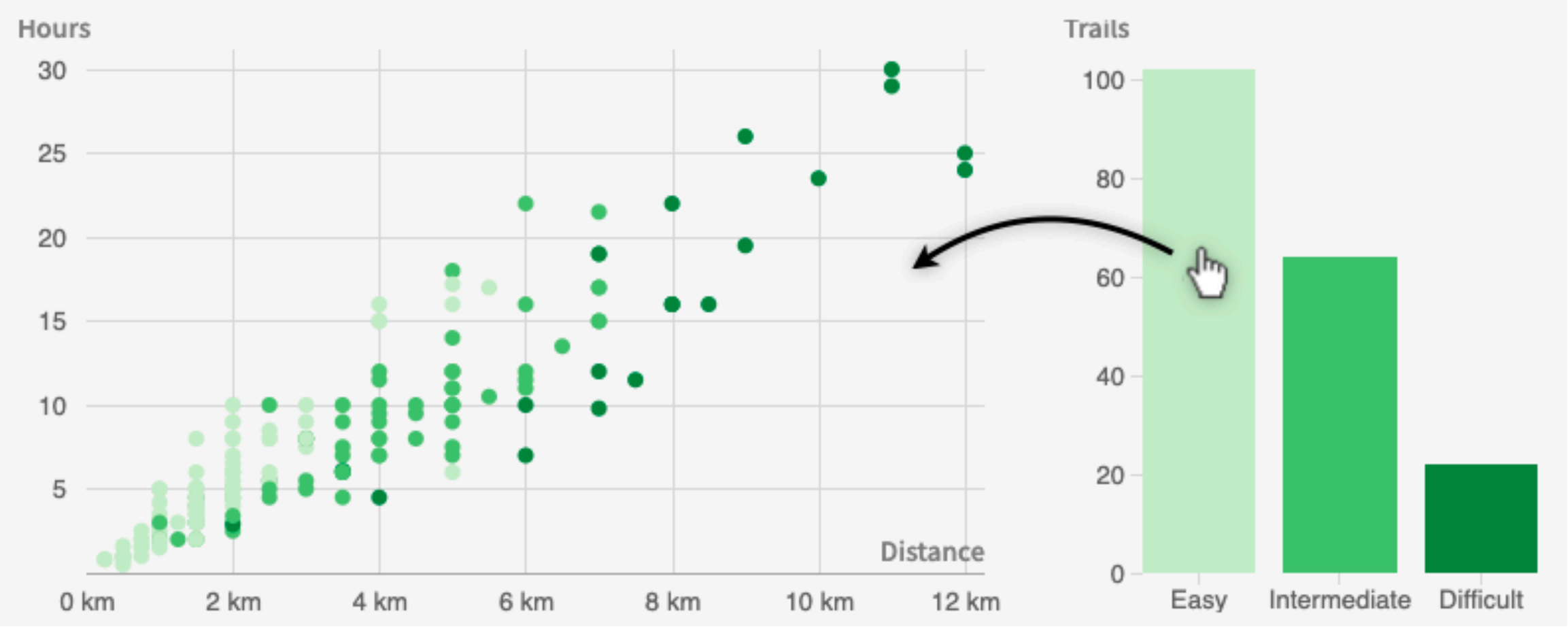
Outline

- Data joins and basic interactivity
- Multiple views and advanced interactivity
- Advanced concepts

Multiple views and advanced interactivity

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

- Linked interactions



Multiple views and advanced interactivity

- Multi-view event handler
  - d3.dispatch()
  - d3.dispatch('filteredCategories', 'selectedPoints', 'reset')

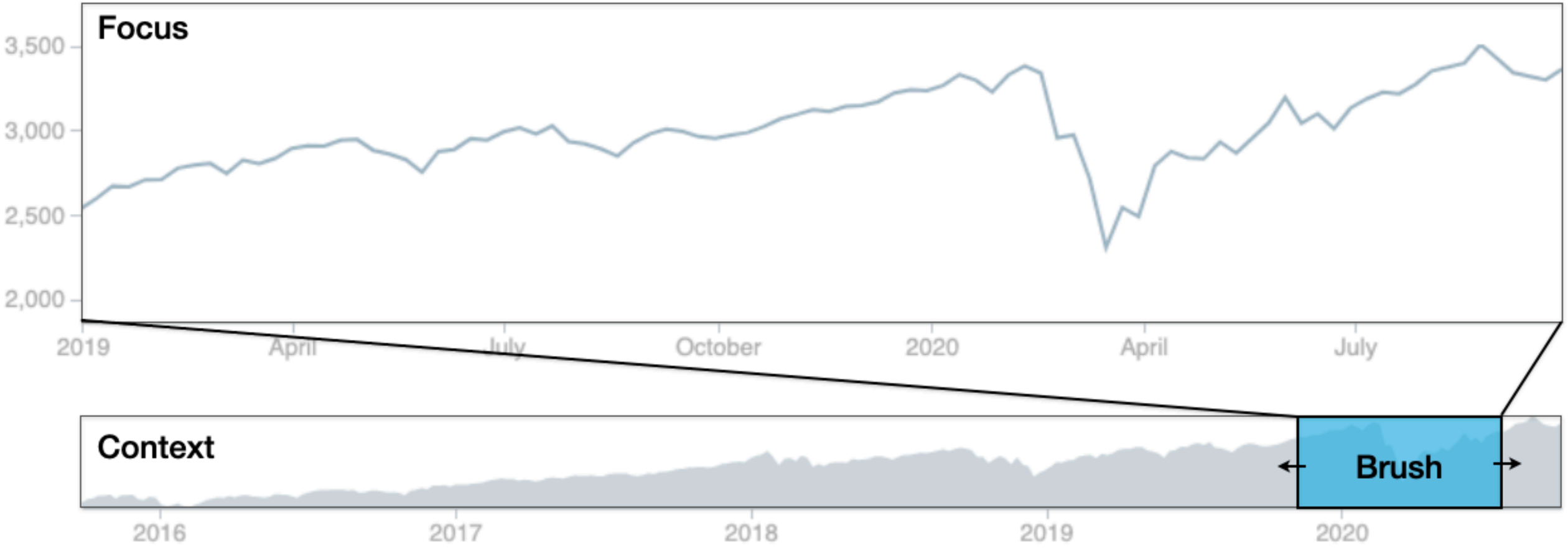
```
dispatcher.on('filterCategories', selectedCategories => {
    if (selectedCategories.length == 0) {
        scatterplot.data = data;
    } else {
        scatterplot.data = data.filter(d => selectedCategories.includes(d.difficulty));
    }
    scatterplot.updateVis();
});
```

Multiple views and advanced interactivity

- Brushing and linking
  - Brushing: a technique to interactively select a region or a set of data points in a visualization
  - linking: change are automatically dispatched to linked visualizations
- Focus + Context
  - The *context view* provides a global perspective at reduced detail and allows users to brush
  - The *focus view* shows the selected data points, for example, a specific time period, in greater detail

Multiple views and advanced interactivity

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen



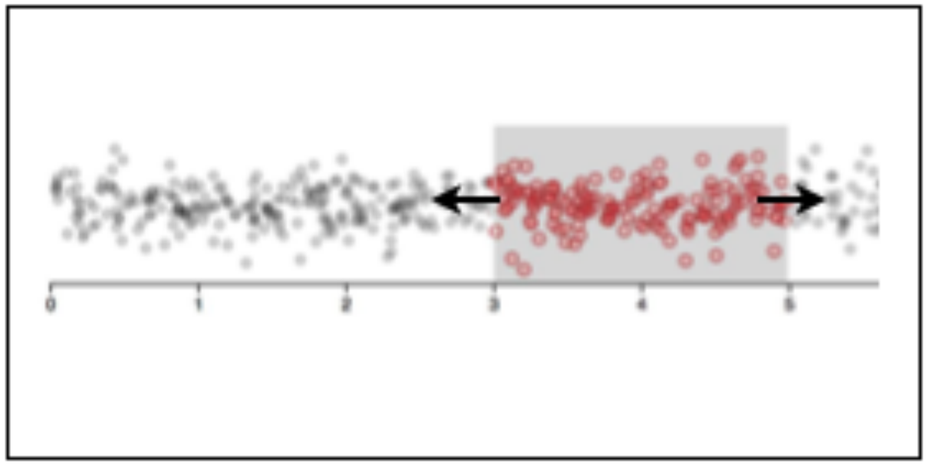
Multiple views and advanced interactivity

- Brushing
  - d3.brushX
  - d3.brushY
  - d3.brush

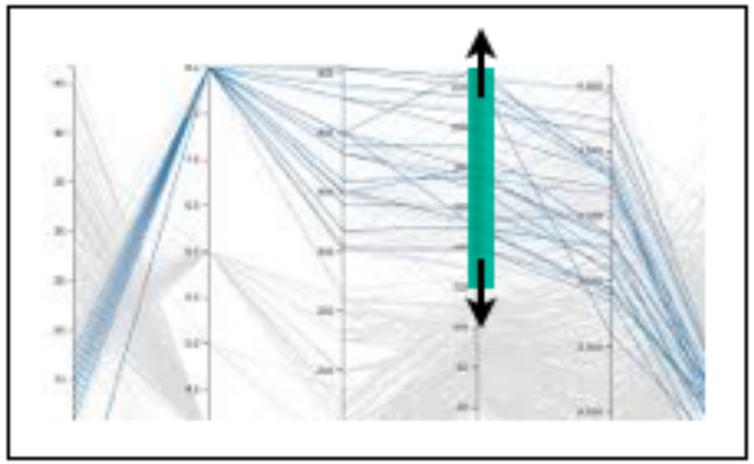
```
const brush = d3.brushX()
    .extent([[0, 0], [width, height]])
    .on('brush', brushed)
    .on('end', brushended);
```

```
const brushG = svg.append('g')
    .attr('class', 'brush x-brush')
    .call(brush);
```

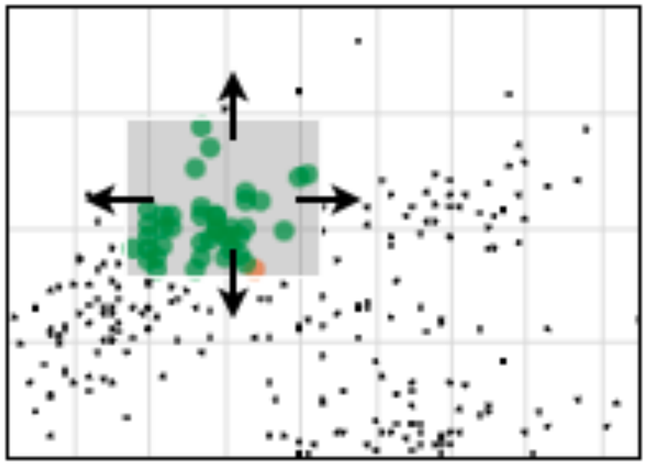
d3.**brushX**



d3.**brushY**



d3.**brush**

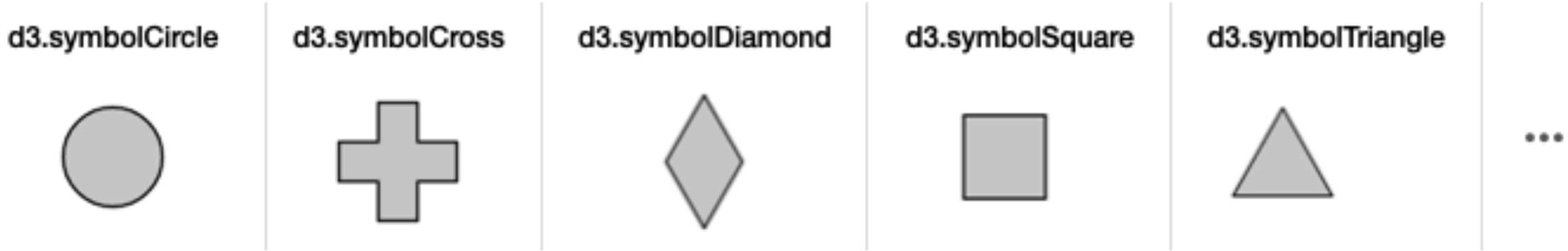


Multiple views and advanced interactivity

- D3 shape generator
  - Line
  - Area
  - Symbol
  - Stacks
  - etc.

Multiple views and advanced interactivity

- Symbol
  - d3.symbol




```
const symbolScale = d3.scaleOrdinal()
    .range([
      d3.symbol().type(d3.symbolCircle)(),
      d3.symbol().type(d3.symbolSquare)(),
      d3.symbol().type(d3.symbolDiamond)()
    ])
    .domain(['Easy', 'Intermediate', 'Difficult']);
```

Multiple views and advanced interactivity

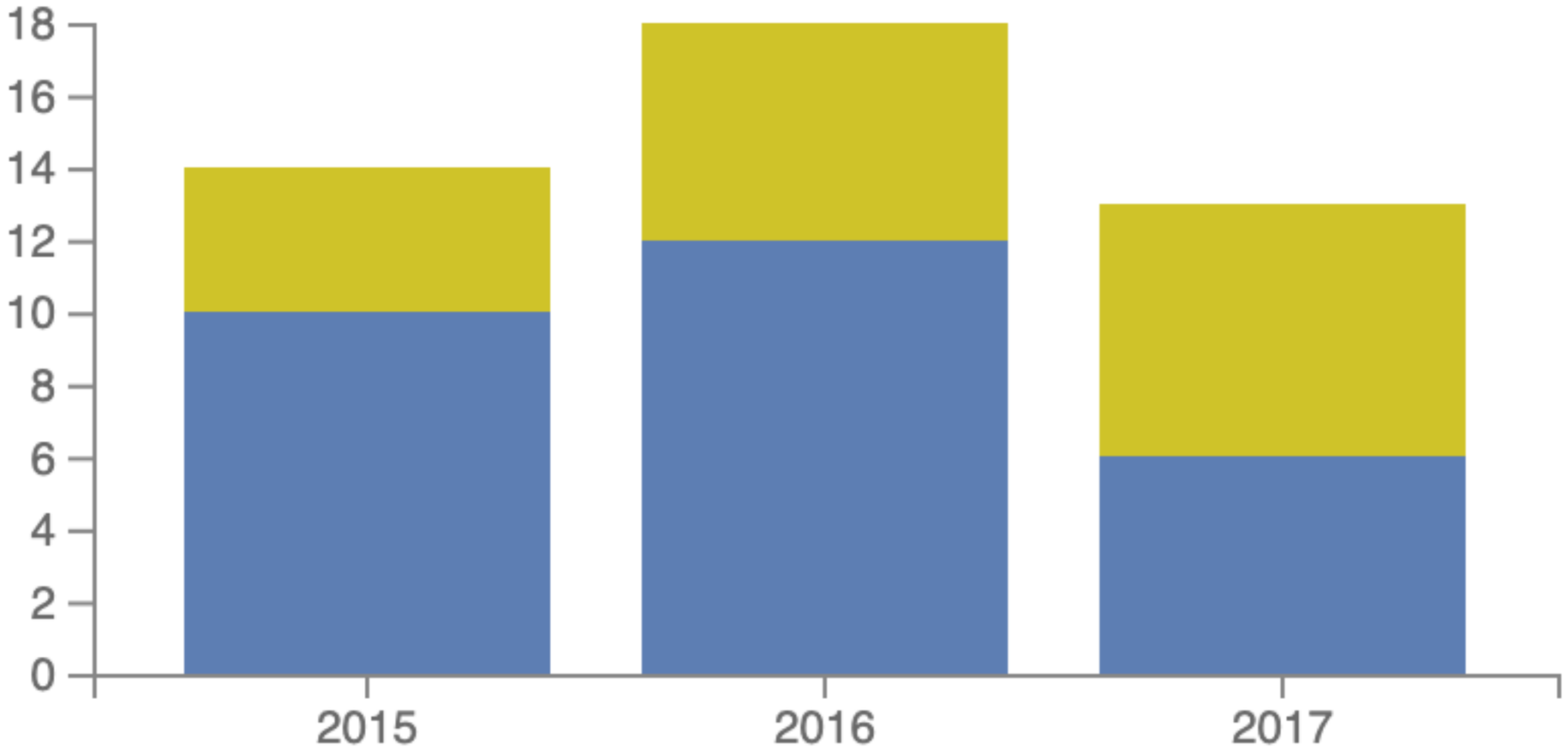
- Stacks
  - d3.stack

```
const data = [
        { 'year': 2015, 'milk': 10, 'water': 4 },
        { 'year': 2016, 'milk': 12, 'water': 6 },
        { 'year': 2017, 'milk': 6, 'water': 7 }
];

const stack = d3.stack().keys(['milk', 'water']);

const stackedData = stack(data);
```

```
const rectangles = svg.selectAll('category')
                .data(stackedData)
        .join('g')
                .attr('class', d => `category cat-${d.key}`)
        .selectAll('rect')
                .data(d => d)
        .join('rect')
                .attr('x', d => xScale(d.data.year))
                .attr('y', d => yScale(d[1]))
                .attr('height', d => yScale(d[0]) - yScale(d[1]))
                .attr('width', xScale.bandwidth());
```

Multiple views and advanced interactivity

- Stacks
  - d3.stack



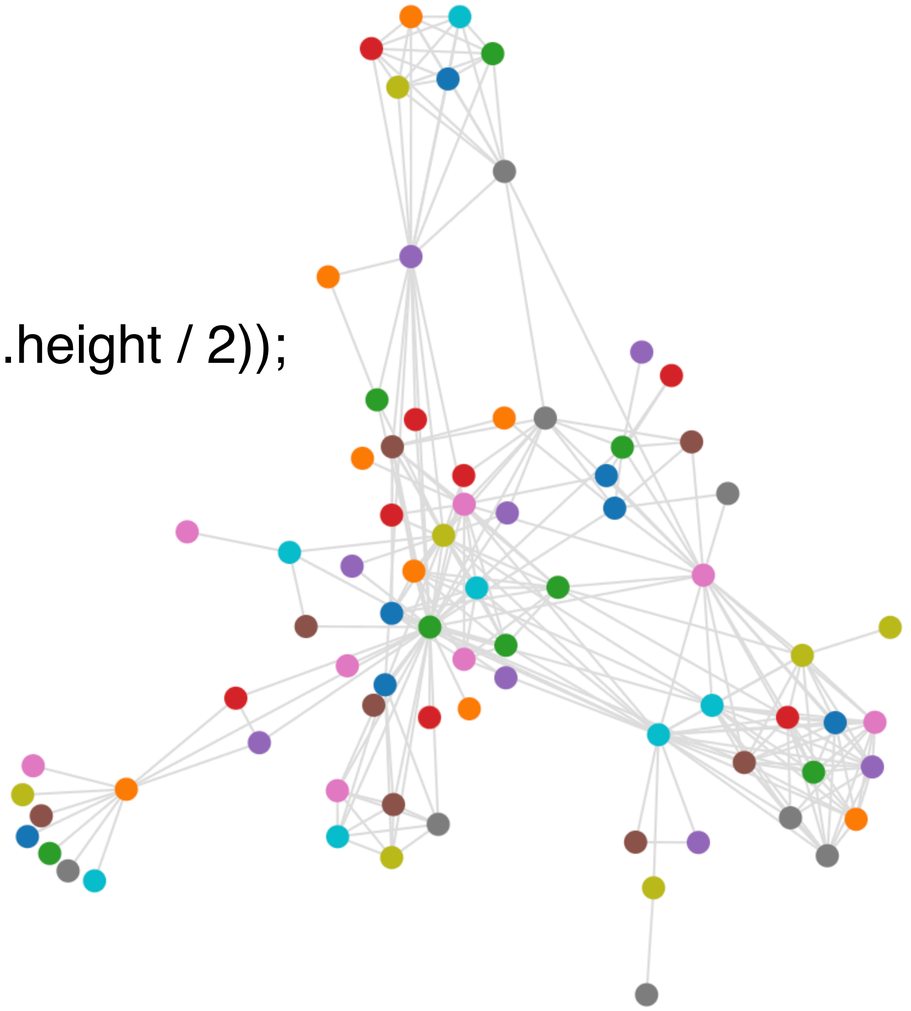
Outline

- Data joins and basic interactivity
- Multiple views and advanced interactivity
- Advanced concepts

Advanced concepts

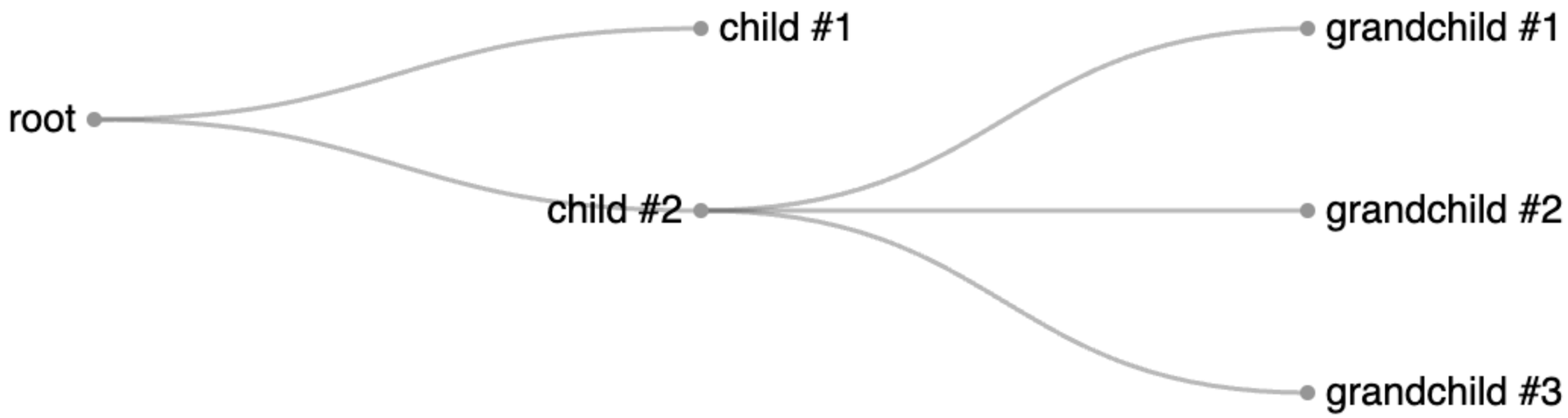
- Graph and trees
  - Graph: forced-directed layout

```
const simulation = d3.forceSimulation()
        .force('link', d3.forceLink().id(d => d.id))
        .force('charge', d3.forceManyBody())
        .force('center', d3.forceCenter(config.width / 2, config.height / 2));
```



- Graph and trees
  - Trees: tiny-tree layout

```
const data = d3.hierarchy(rawData)
const treeData = d3.tree().size([height, width])(data);
```

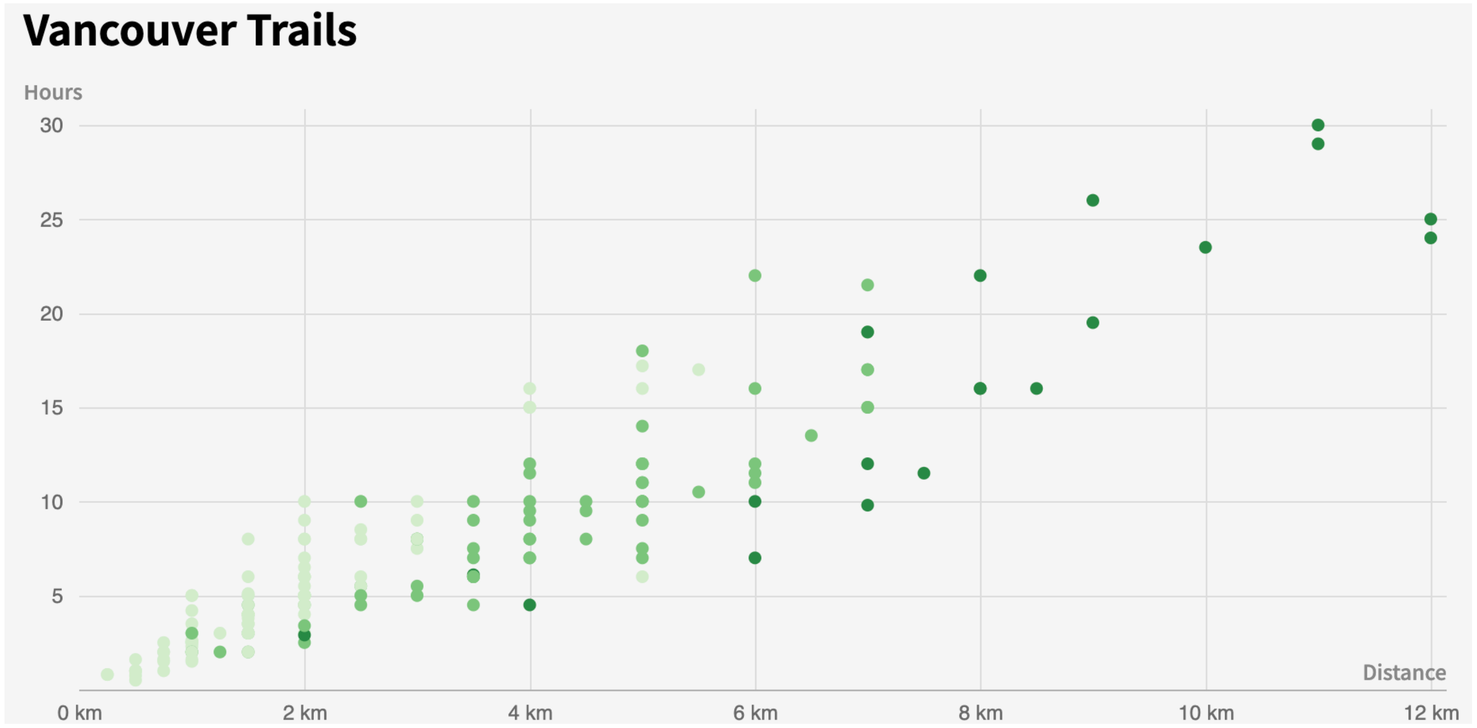


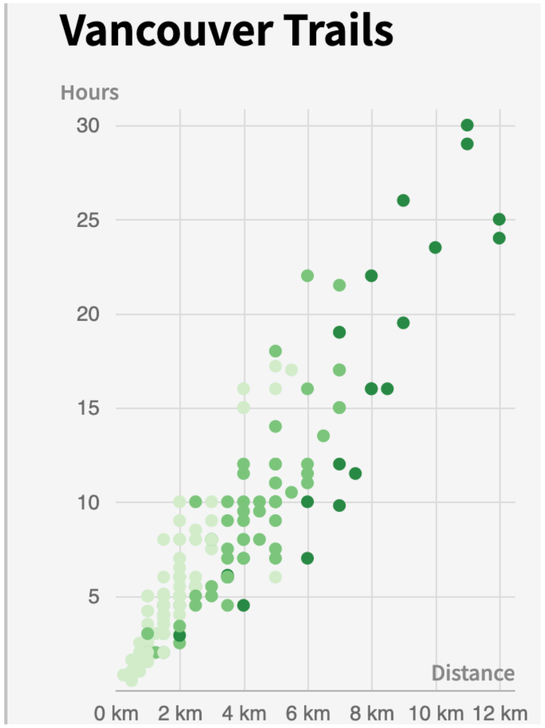
- Scroll-based interaction
  - A popular technique to reveal or alter content based on the user's scroll behavior
  - Recommend Waypoints to listen for scroll events
  - https://www.theguardian.com/us-news/ng-interactive/2015/oct/19/homan-square-chicago-police-detainees
  - https://pudding.cool/2017/03/hamilton/index.html
  - https://vallandingham.me/scroller.html

Advanced concepts

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

- Responsive visualization
  - Dynamically change the size of visualization forms based on wide variety of screen sizes





- Annotations
  - Make a visualization more accessible to your target audience
  - Improve clarity, for example, by explaining data anomalies or highlighting interesting patterns in a visualization
  - https://d3-annotation.susielu.com/

Thank Dr. Michael Oppermann for many of the slides!