D3 Exercises & Questioning

Presenter: Shiwangi Singh

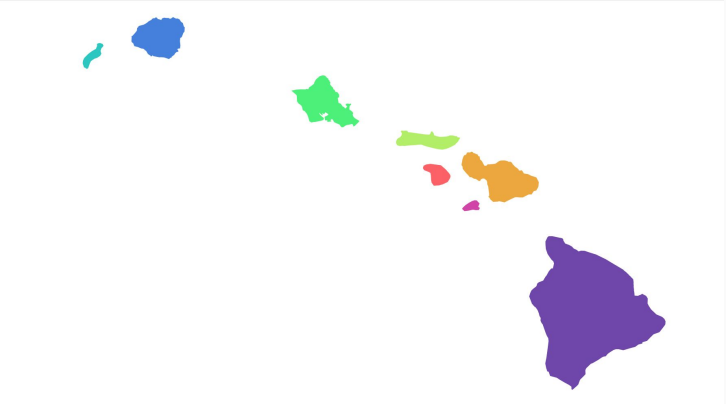
Discussion Topics

Questions :

- Update Pattern
- Merge and Exit

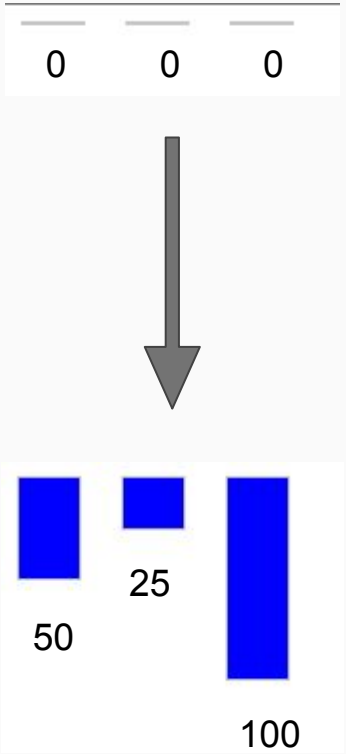
D3 Examples :

- Smooth Transitioning
- Voronoi Topology
- Jigsaw Morphing



Update Patterns

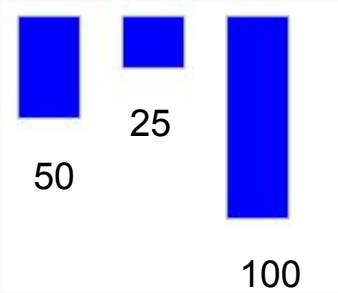
```
<html>
▶#shadow-root (open)
▶<head>…</head>
▼<body>
  ▼<div id="chart">
      <div class="bar" style="height: 50px; background-color: blue;"></div>
      <div class="bar" style="height: 25px; background-color: blue;"></div>
      <div class="bar" style="height: 100px; background-color: blue;"></div>
    </div>
    <script type="text/javascript" src="d3.min.js"></script>
  </body>
</html>
```

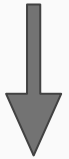



```
d3.select('#chart').selectAll('.bar')
        .data([50,25,100])
        .style('height', d=> d + 'px')
        .style('background-color', 'blue')
```

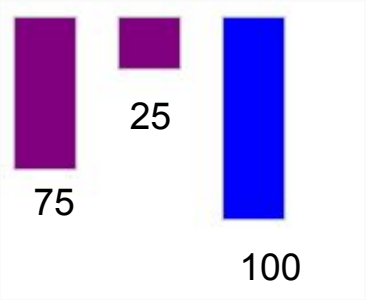
Update Patterns

```
<html>
▶#shadow-root (open)
▶<head>…</head>
▼<body>
  ▼<div id="chart">
      <div class="bar" style="height: 75px; background-color: purple;"></div>
      <div class="bar" style="height: 25px; background-color: purple;"></div>
      <div class="bar" style="height: 100px; background-color: blue;"></div>
    </div>
    <script type="text/javascript" src="d3.min.js"></script>
  </body>
</html>
```



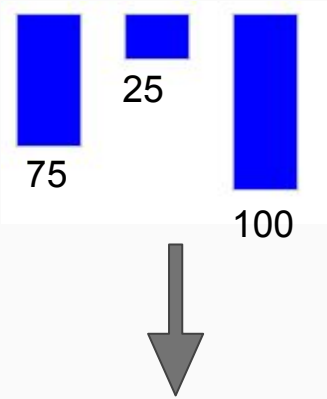


```
d3.select('#chart').selectAll('.bar')
        .data([75,25])
        .style('height', d=> d + 'px')
        .style('background-color', 'purple')
```

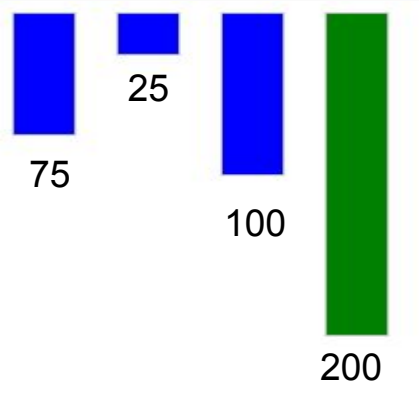


“Enter” Selection

```
<html>
▶#shadow-root (open)
▶<head>…</head>
▼<body>
  ▼<div id="chart">
      <div class="bar" style="height: 75px; background-color: blue;"></div>
      <div class="bar" style="height: 25px; background-color: blue;"></div>
      <div class="bar" style="height: 100px; background-color: blue;"></div>
      <div class="bar" style="height: 200px; background-color: green;"></div>
    </div>
    <script type="text/javascript" src="d3.min.js"></script>
  </body>
</html>
```

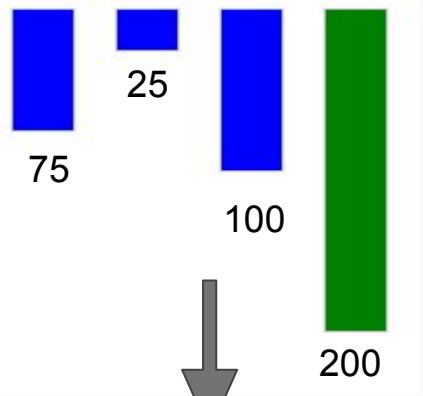



```
d3.select(‘#chart’).selectAll(‘.bar’)
        .data([75,25,100,200])
        .style(‘height’, d=> d + ‘px’)
        .style(‘background-color’, ‘blue’)
        .enter()
        .append(‘div’)
          .attr(‘class’, ‘bar’)
          .style(‘height’, d=> d + ‘px’)
          .style(‘background-color’, ‘green’)
```

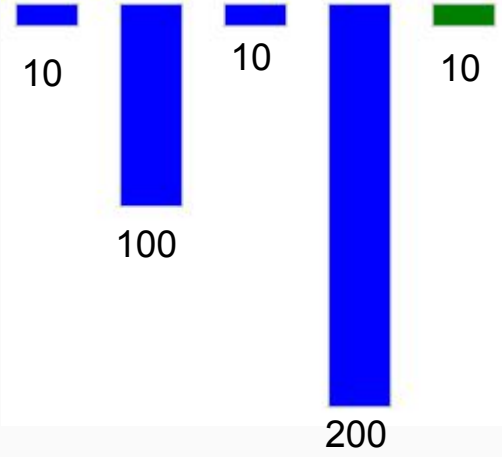


Merge: “Update + Enter”

```
<html>
▶#shadow-root (open)
▶<head>…</head>
▼<body>
  ▼<div id="chart">
      <div class="bar" style="height: 75px; background-color: blue;"></div>
      <div class="bar" style="height: 25px; background-color: blue;"></div>
      <div class="bar" style="height: 100px; background-color: blue;"></div>
      <div class="bar" style="height: 200px; background-color: green;"></div>
    </div>
    <script type="text/javascript" src="d3.min.js"></script>
  </body>
</html>
```



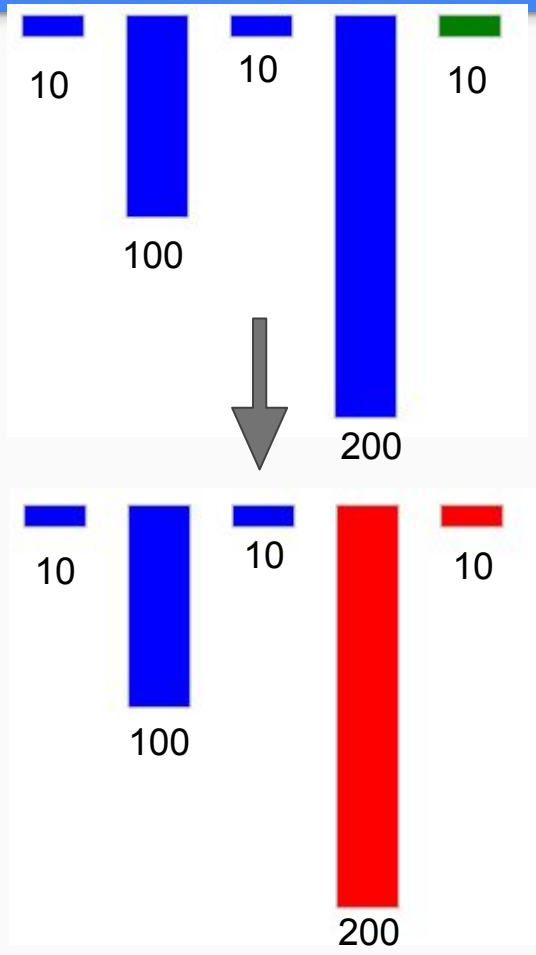

```
d3.select(‘#chart’).selectAll(‘.bar’)
        .data([10,100,10,200,10])
        .style(‘background-color’, ‘blue’)
        .enter()
        .append(‘div’)
          .attr(‘class’, ‘bar’)
          .style(‘background-color’, ‘green’)
       .merge(d3.select(‘#chart’).selectAll(‘.bar’))
          .style(‘height’, d=> d + ‘px’)
```



Exit

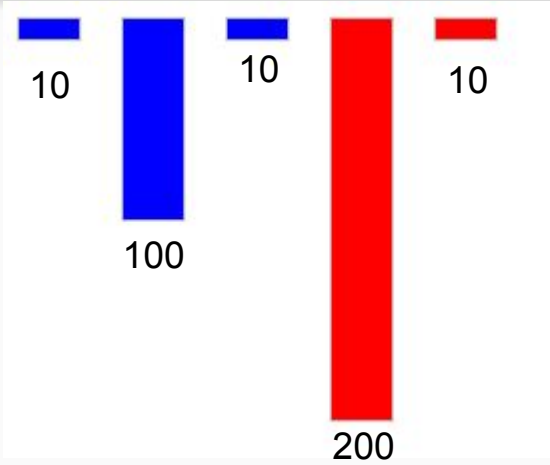
```
<html>
▶ #shadow-root (open)
▶ <head>…</head>
▼ <body>
  ▼ <div id="chart">
      <div class="bar" style="background-color: blue; height: 10px;"></div>
      <div class="bar" style="background-color: blue; height: 100px;"></div>
      <div class="bar" style="background-color: blue; height: 10px;"></div>
      <div class="bar" style="background-color: red; height: 200px;"></div>
      <div class="bar" style="background-color: red; height: 10px;"></div>
    </div>
    <script type="text/javascript" src="d3.min.js"></script>
  </body>
</html>
```

d3.**select**(‘#chart’).**selectAll**(‘.bar’)
.**data**([10,100,10])
.**style**(‘background-color’, ‘blue’)
.exit()
.**style**(‘background-color’, red’)

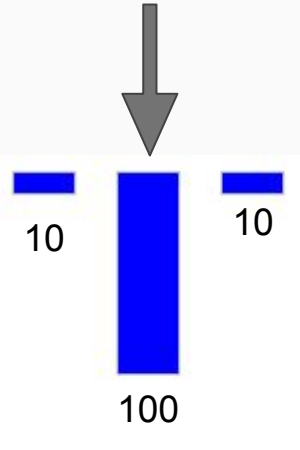


Exit

```
<html>
▶#shadow-root (open)
▶<head>…</head>
▼<body>
  ▼<div id="chart">
      <div class="bar" style="background-color: blue; height: 10px;"></div>
      <div class="bar" style="background-color: blue; height: 100px;"></div>
      <div class="bar" style="background-color: blue; height: 10px;"></div>
    </div>
    <script type="text/javascript" src="d3.min.js"></script>
  </body>
</html>
```



d3.**select**(‘#chart’).**selectAll**(‘.bar’)
.**data**([10,100,10])
.**style**(‘background-color’, ‘blue’)
.exit()
.**style**(‘background-color’, red’)
.remove()



Putting it all together

```
function Update(data) {
    let barsUpdate = d3.select('#chart').selectAll('.bar').data(data)
    let barsEnter = barsUpdate.enter()
    let barsExit = barsUpdate.exit()

    barsUpdate
        .style('background-color', 'blue')
        .style('height', d => d+'px')

    barsEnter
        .append('div').attr('class', 'bar')
        .style('background-color', 'green')
        .style('height', d => d+'px')

    barsExit
        .style('background-color', 'red')
        .remove()  // in this case styling would be pointless
}
```

All three patterns can 'coexist' in the same function or scope.

Modify EXISTING elements

Appends new data to DOM. Affects only NEW DATA

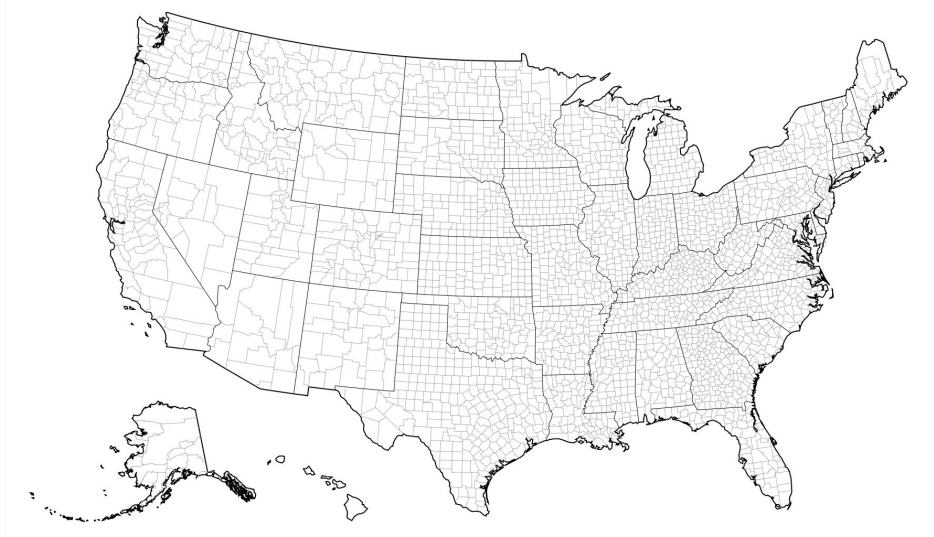
Modify (or remove) elements with MISSING data, or data that has exited the selection

TopoJSON

TopoJSON is an extension of GeoJSON that encodes Topology.

GeoJSON - Represents geometries discretely (e.g Polygon, Multipolygon, etc)

TopoJSON - Has fixed-precision integer encoding for co-ordinates
80 % smaller than GeoJSON file
Maps back to GeoJSON



TopoJSON Exercise

Create a US states TopoJSON map

Go to Piazza under In-class exercise 09/27 :

- Grab the html, js and JSON codes from under Topomap heading
- Create a folder called topomap and put the above files into it
- Navigating to the current dir, run a local server
  - In python2.7: `python -m SimpleHTTPServer $port`
  - In python3.x: `python -m http.server $port`

By the end of this exercise, you know :

- How to create a us state map using TopoJSON
- To explore the JSON data file to see arcs and coordinates in topojson

Those who already know how to do this:

See Smooth Polygon Transition example on bl.ocks.org

Example 1 : Smooth Polygon Transitions

Go to *Smooth Polygon Transition example on bl.ocks.org*

```
// Redraw points
circles.datum(a)
  .call(updateCircles);

// Morph
var t = d3.transition()
  .duration(800);

path.transition(t)
  .on("end", function(){
    states.push(states.shift());
    setTimeout(draw, 100);
  })
  .attr("d", join(b));

circles.selectAll("circle").data(b)
  .transition(t)
  .attr("cx",function(d){
    return d[0];
  })
  .attr("cy",function(d){
    return d[1];
  });
```

Data Join

(state a, state b)
Pop a
(now a = b)
Push new state as b

```
function updateCircles(sel) {

  var circles = sel.selectAll("circle")
    .data(function(d){ return d; });

  var merged = circles.enter()
    .append("circle")
    .attr("r", 2)
    .merge(circles);

  merged.classed("added", function(d){
      return d.added;
    })
    .attr("cx",function(d){
      return d[0];
    })
    .attr("cy",function(d){
      return d[1];
    });

  circles.exit().remove();
}
```

Merge new a

added class as pink

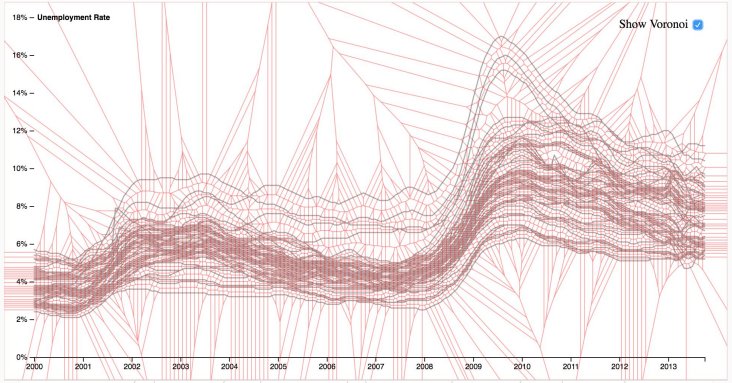
Remove state a circles

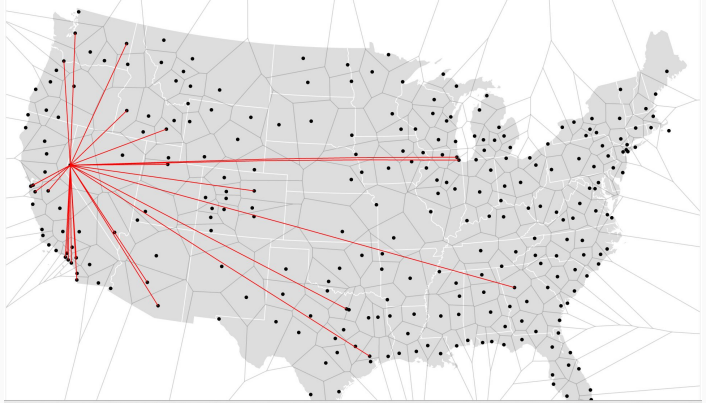
Voronoi Diagrams

What are Voronoi Diagrams ?

Wikipedia definition says “A Voronoi Diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane”

- Set of points is called sites or seeds.
- Each region is called a Voronoi Cell.
- Based on Euclidean distance or Manhattan distance.





What is the use of Voronoi Diagram ?

- To increase the Target area of points in a scatterplot
- For interactions
- Computing adjacency or grouping of Visual elements
- Automate label positioning

Voronoi Exercise

Create a Voronoi Diagram with random data

Go to Piazza under In-class exercise 09/27 :

- Grab the html, js and JSON codes from under Voronoi heading
- Create a folder called voronoi and put the above files into it
- Navigating to the current dir, run a local server
  - In python2.7: python -m SimpleHTTPServer $port
  - In python3.x: python -m http.server $port

By the end of this exercise, you should know :

- How to create a Voronoi diagram using d3.voronoi api
- How to do basic interaction with voronoi diagram

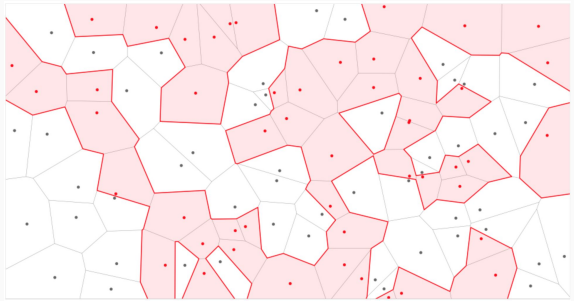
If you are done with this exercise :

See Voronoi Topology on bl.ocks.org

Example 2 : Voronoi Topology

See Voronoi Topology on bl.ocks.org

- This Example makes use of both the Topo map and Voronoi diagram
- Shows interaction with Voronoi cells
- Makes use of two important features topojson.merge and topojson.mesh
- This example uses Canvas element rather than SVG



Q. *What is the difference between Canvas and SVG ?*

SVG is shape-based which are comprised of DOM elements, each element can be modified
Canvas is pixel-based which is like an image, entire canvas is rendered if modified

Example 2 : Voronoi Topology

```
var voronoi = d3.voronoi()
    .extent([[-1, -1], [width + 1, height + 1]]);
```

Makes a voronoi diagram of width and height specified

```
var timer = d3.timer(function(elapsed) {
  for (var i = 0; i < n; ++i) {
    var p = particles[i];
    p[0] += p.vx; if (p[0] < 0) p[0] = p.vx *= -1; else if (p[0] > width) p[0] = width + (p.vx *= -1
    p[1] += p.vy; if (p[1] < 0) p[1] = p.vy *= -1; else if (p[1] > height) p[1] = height + (p.vy *=
    p.vx += 0.1 * (Math.random() - 0.5) - 0.01 * p.vx;
    p.vy += 0.1 * (Math.random() - 0.5) - 0.01 * p.vy;
  }
```

Assigns random coordinates (x, y) within the specified dimension

```
var topology = computeTopology(voronoi(particles));

context.clearRect(0, 0, width, height);
```

voronoi (data) computes V-diagram for ‘data’. New voronoi is computed.

Example 2 : Voronoi Topology

```
return {
  objects: {
    voronoi: {
      type: "GeometryCollection",
      geometries: cells.map(function(cell) {
        var cell,
            site = cell.site,
            halfedges = cell.halfedges,
            cellArcs = [],
            clipArc;

        halfedges.forEach(function(halfedge) {
          var edge = diagram.edges[halfedge];
          if (edge.right) {
            var l = edge.left.index,
                r = edge.right.index,
                k = l + "," + r,
                i = arcIndexByEdge[k];
            if (i == null) arcs[i = arcIndexByEdge[k] = ++arcIndex] = edge;
            cellArcs.push(site === edge.left ? i : ~i);
            clipArc = null;
          } else if (clipArc) { // Coalesce border edges.
            if (edge.left) edge = edge.slice(); // Copy-on-write.
            clipArc.push(edge[1]);
          } else {
            arcs[++arcIndex] = clipArc = edge;
            cellArcs.push(arcIndex);
          }
        });
```

Entire topology is retuned

Voronoi has (cells, edges)
Cells have (site, halfedges)
Site has (index, data)
Edge has (left, right)

New cell arcs are computed based on edge right or left for each Cell

Example 2 : Voronoi Topology

What are topojson.merge and topojson.mesh ?

topojson.**mesh**(*topology*, *object*, [*filter*])

Returns the GeoJSON MultiLineString geometry object representing the mesh for the specified *object* in the given *topology*. This is useful for rendering strokes in complicated objects efficiently, as edges that are shared by multiple features are only stroked once.

topojson.**merge**(*topology*, *objects*)

Returns the GeoJSON MultiPolygon geometry object representing the union for the specified array of Polygon and MultiPolygon *objects* in the given *topology*. Interior borders shared by adjacent polygons are removed. See Merging States for an example.

Example 3 : Jigsaw Morphing

See Jigsaw morphing on bl.ocks.org