07. D3JS Drawing Charts.

D3.js is used to create a static SVG chart. It helps to draw the following charts:

- **Bar Chart**
- **Circle Chart**
- **Pie Chart**
- **Donut Chart**
- **Line Chart**
- **Bubble Chart**
- **...**

This chapter explains about drawing charts in D3. Let us understand each of these in detail.

7.1. Bar Chart

Bar charts are one of the most commonly used types of graph and are used to display and compare the number, frequency or other measure (e.g. mean) for different discrete categories or groups. This graph is constructed in such a way that the heights or lengths of the different bars are proportional to the size of the category they represent.

The x-axis (the horizontal axis) represents the different categories it has no scale. The y axis (the vertical axis) does have a scale and this indicates the units of measurement. The bars can be drawn either vertically or horizontally depending upon the number of categories and length or complexity of the category.

Draw a Bar Chart

Let us create a bar chart in SVG using D3. For this example, we can use the rect elements for the bars and text elements to display our data values corresponding to the bars.

To create a bar chart in SVG using D3, let us follow the steps given below.

Step 1: Adding style in the rect element − Let us add the following style to the rect element.

```
svg rect {
   fill: gray;
}
```

Step 2: Add styles in text element − Add the following CSS class to apply styles to text values. Add this style to SVG text element. It is defined below

```
svg text {
   fill: yellow;
   font: 12px sans-serif;
   text-anchor: end;
}
```

Here, Fill is used to apply colors. Text-anchor is used to position the text towards the right end of the bars.

Step 3: Define variables − Let us add the variables in the script. It is explained below.

```
<script>
   var data = [10, 5, 12, 15];
   var width = 300,
      scaleFactor = 20,
      barHeight = 30;
</script>
```

Here,

- **Width** − Width of the SVG.

- **Scalefactor** − Scaled to a pixel value that is visible on the screen.
- **Barheight** − This is the static height of the horizontal bars.

Step 4: Append SVG elements − Let us append SVG elements in D3 using the following code.

```
var graph = d3.select("body")
   .append("svg")
   .attr("width", width)
   .attr("height", barHeight * data.length);
```

Here, we will first select the document body, create a new SVG element and then append it. We will build our bar graph inside this SVG element. Then, set the width and height of SVG. Height is calculated as bar height * number of data values.

We have taken the bar height as 30 and data array length is 4. Then SVG height is calculated as barheight* datalength which is 120 px.

Step 5: Apply transformation − Let us apply the transformation in bar using the following code.

```
var bar = graph.selectAll("g")
   .data(data)
   .enter()
   .append("g")
   .attr("transform", function(d, i) {
      return "translate(0," + i * barHeight + ")";
   });
```

Here, each bar inside corresponds with an element. Therefore, we create group elements. Each of our group elements needs to be positioned one below the other to build a horizontal bar chart. Let us take a transformation formula index * bar height.

Step 6: Append rect elements to the bar − In the previous step, we appended group elements. Now add the rect elements to the bar using the following code.

```
bar.append("rect")
   .attr("width", function(d) {
      return d * scaleFactor;
   })
   .attr("height", barHeight - 1);
```

Here, the width is (data value * scale factor) and height is (bar height – margin).

Step 7: Display data − This is the last step. Let us display the data on each bar using the following code.

```
bar.append("text")
   .attr("x", function(d) { return (d*scaleFactor); })
   .attr("y", barHeight / 2)
   .attr("dy", ".35em")
   .text(function(d) { return d; });
```

Here, width is defined as (data value * scalefactor). Text elements do not support padding or margin. For this reason, we need to give it a "dy" offset. This is used to align the text vertically.

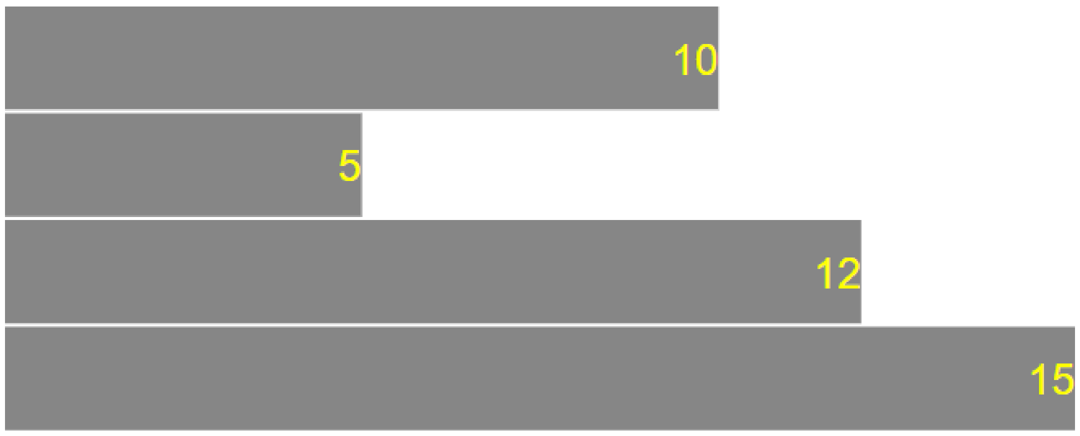
Step 8: Working example − The complete code listing is shown in the following code block. Create a webpage barcharts.html and add the following changes.

```
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
      <style>
         svg rect {
            fill: gray;
         }

         svg text {
            fill: yellow;
            font: 12px sans-serif;
            text-anchor: end;
         }
      </style>
   </head>

   <body>
      <script>
         var data = [10, 5, 12, 15];

         var width = 300
            scaleFactor = 20,
            barHeight = 30;

         var graph = d3.select("body")
            .append("svg")
            .attr("width", width)
            .attr("height", barHeight * data.length);

         var bar = graph.selectAll("g")
            .data(data)
            .enter()
            .append("g")
            .attr("transform", function(d, i) {
               return "translate(0," + i * barHeight + ")";
            });
```

```
               bar.append("rect").attr("width", function(d) {
                  return d * scaleFactor;
               })

               .attr("height", barHeight - 1);

               bar.append("text")
                  .attr("x", function(d) { return (d*scaleFactor); })
                  .attr("y", barHeight / 2)
                  .attr("dy", ".35em")
                  .text(function(d) { return d; });
            </script>
         </body>
      </html>
```

Now request your browser, you will see the following response.

Output:



7.2. Circle Chart

A Circle chart is a circular statistical graphic, which is divided into slices to illustrate a numerical proportion.

Draw a Circle Chart

Let us create a circle chart in SVG using D3. To do this, we must adhere to the following steps −

Step 1: Define variables - Let us add the variables in the script. It is shown in the code block below.

```
<script>
   var width = 400;
   var height = 400;
   var data = [10, 20, 30];
   var colors = ['green', 'purple', 'yellow'];
</script>
```

Here,

- Width − width of the SVG.
- Height − height of the SVG.
- Data − array of data elements.
- Colors − apply colors to the circle elements.

Step 2: Append SVG elements - Let us append SVG elements in D3 using the following code.

```
var svg = d3.select("body")
   .append("svg")
   .attr("width", width)
   .attr("height", height);
```

Step 3: Apply transformation − Let us apply the transformation in SVG using the following code.

```
var g = svg.selectAll("g")
   .data(data)
   .enter()
   .append("g")
   .attr("transform", function(d, i) {
      return "translate(0,0)";
   })
```

Here,

- var g = svg.selectAll("g") − Creates group element to hold the circles.
- .data(data) − Binds our data array to the group elements.
- .enter() − Creates placeholders for our group elements.
- .append("g") − Appends group elements to our page.
- Translate(0,0) - used to position your elements with respect to the origin.

Step 4: Append circle elements − Now, append circle elements to the group using the following code.

```
g.append("circle")
```

Example: Now, add the attributes to the group using the following code.

```
.attr("cx", function(d, i) {
   return i*75 + 50;
})
```

Here, we use the x-coordinate of the center of each circle. We are multiplying the index of the circle with 75 and adding a padding of 50 between the circles.

Next, we set the y-coordinate of the center of each circle. This is used to uniform all the circles and it is defined below.

Example:

```
.attr("cy", function(d, i) {
   return 75;
})
```

Next, set the radius of each circle. It is defined below,

Example:

```
.attr("r", function(d) {
   return d*1.5;
})
```

Here, the radius is multiplied with data value along with a constant "1.5" to increase the circle's size. Finally, fill colors for each circle using the following code.

Example:

```
.attr("fill", function(d, i){
   return colors[i];
})
```

Step 5: Display data − This is the last step. Let us display the data on each circle using the following code.

```
g.append("text")
   .attr("x", function(d, i) {
      return i * 75 + 25;
   })
   .attr("y", 80)
   .attr("stroke", "teal")
   .attr("font-size", "10px")
   .attr("font-family", "sans-serif")
   .text(function(d) {
      return d;
   });
```

Step 6: Working example − The complete code listing is shown in the following code block. Create a webpage circlecharts.html and add the following changes in it.

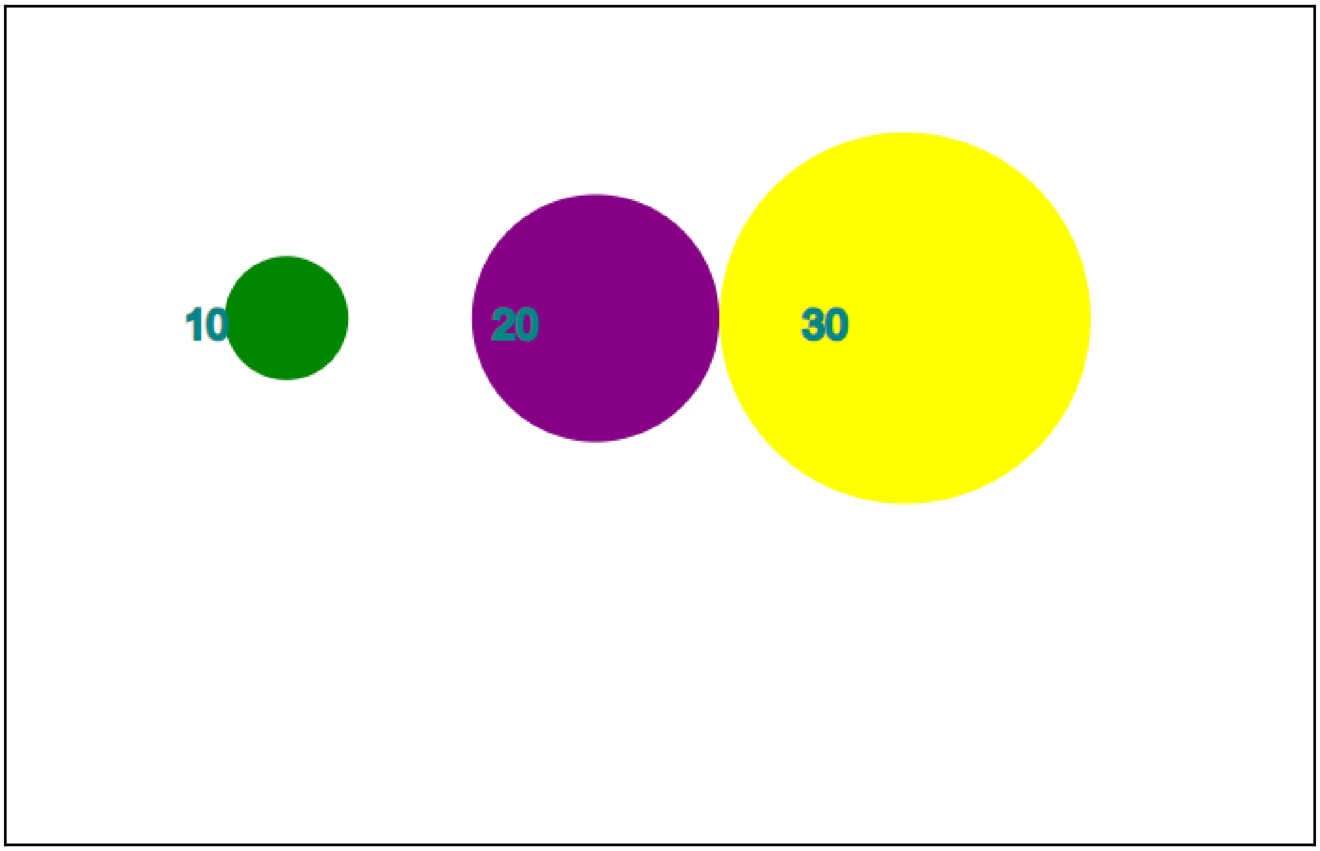
```
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
   </head>

   <body>
      <script>
         var width = 400;

         var height = 400;

         var data = [10, 20, 30];

         var colors = ['green', 'purple', 'yellow'];

         var svg = d3
```

```
        .select("body")
        .append("svg")
        .attr("width", width)
        .attr("height", height);

    var g = svg.selectAll("g")
        .data(data)
        .enter()
        .append("g")
        .attr("transform", function(d, i) {
            return "translate(0,0)";
        })

    g.append("circle").attr("cx", function(d, i) {
        return i*75 + 50;
    })

    .attr("cy", function(d, i) {
        return 75;
    })

    .attr("r", function(d) {
        return d*1.5;
    })

    .attr("fill", function(d, i){
        return colors[i];
    })

    g.append("text").attr("x", function(d, i) {
        return i * 75 + 25;
    })

    .attr("y", 80)
    .attr("stroke", "teal")
    .attr("font-size", "10px")
    .attr("font-family", "sans-serif").text(function(d) {
```

```
                  return d;
               });
            </script>
         </body>
      </html>
```

Now, request your browser and following will be the response.

Output:



7.3. Pie Chart

A pie chart is a circular statistical graph. It is divided into slices to show numerical proportion. Let us understand how to create a pie chart in D3.

Draw a Pie Chart

Before starting to draw a pie chart, we need to understand the following two methods −

- The d3.arc() method and
- The d3.pie() method.

Let us understand both of these methods in detail.

- **The d3.arc() Method** − The d3.arc() method generates an arc. You need to set an inner radius and an outer radius for the arc. If the inner radius is 0, the result will be a pie chart, otherwise the result will be a donut chart, which is discussed in the next section.

- **The d3.pie()Method** − The d3.pie() method is used to generate a pie chart. It takes a data from dataset and calculates the start angle and end angle for each wedge of the pie chart.

Let us draw a pie chart using the following steps.

Step 1: Applying styles − Let us apply the following style to an arc element.

```
.arc text {
   font: 12px arial;
   text-anchor: middle;
}

.arc path {
   stroke: #fff;
}

.title {
   fill: green;
   font-weight: italic;
}
```

Here, fill is used to apply colors. A text-anchor is used to position the text towards the center of an arc.

Step 2: Define variables − Define the variables in the script as shown below.

```
<script>
   var svg = d3.select("svg"),
      width = svg.attr("width"),
      height = svg.attr("height"),
      radius = Math.min(width, height) / 2;
</script>
```

Here,

- Width − Width of the SVG.
- Height − Height of the SVG.
- Radius − It can be calculated using the function of Math.min(width, height) / 2;

Step 3: Apply Transformation − Apply the following transformation in SVG using the following code.

```
var g = svg.append("g")
   .attr("transform",
         "translate(" + width / 2 + "," + height / 2 + ")"
);
```

Now add colors using the ***d3.scaleOrdinal*** function as given below.

Example:

```
var color = d3.scaleOrdinal(['gray', 'green', 'brown', 'orange']);
```

Step 4: Generate a pie chart − Now, generate a pie chart using the function given below.

```
var pie = d3.pie()
   .value(function(d) { return d.percent; });
```

Here, you can plot the percentage values. An anonymous function is required to return d.percent and set it as the pie value.

Step 5: Define arcs for pie wedges − After generating the pie chart, now define arc for each pie wedges using the function given below.

```
var arc = d3.arc()
   .outerRadius(radius)
   .innerRadius(0);
```

Step 6: Add labels in wedges − Add the labels in pie wedges by providing the radius. It is defined as follows.

```
var label = d3
   .arc()
   .outerRadius(radius)
   .innerRadius(radius - 80);
```

Step 7: Read data − This is an important step. We can read data using the function given below.

```
d3.csv("populations.csv", function(error, data) {
   if (error) {
      throw error;
   }
});
```

Here, populations.csv contains the data file. The d3.csv function reads data from the dataset. If data is not present, it throws an error. We can include this file in your D3 path.

Step 8: Load data − The next step is to load data using the following code.

```
var arc = g.selectAll(".arc")
   .data(pie(data))
   .enter()
   .append("g")
   .attr("class", "arc");
```

Here, we can assign data to group elements for each of the data values from the dataset.

Step 9: Append path – Now, append path and assign a class 'arc' to groups as shown below.

```
arcs.append("path")
   .attr("d", arc)
   .attr("fill", function(d) { return color(d.data.states); });
```

Here, fill is used to apply the data color. It is taken from the d3.scaleOrdinal function.

Step 10: Append text – To display the text in labels using the following code.

```
arc.append("text")
   .attr("transform", function(d) {
      return "translate(" + label.centroid(d) + ")";
   })
.text(function(d) { return d.data.states; });
```

Here, SVG text element is used to display text in labels. The label arcs that we created earlier using d3.arc() returns a centroid point, which is a position for labels. Finally, we provide data using the d.data.browser.

Step 11: Append group elements – Append group elements attributes and add class title to color the text and make it italic, which is specified in Step 1 and is defined below.

```
svg.append("g")
   .attr("transform",
         "translate(" + (width / 2 - 120) + "," + 20 + ")")
   .append("text")
   .text("Top population states in india")
   .attr("class", "title")
```

Step 12: Working example – To draw a pie chart, we can take a dataset of Indian population. This dataset shows the population in a dummy website, which is defined as follows.

File "population.csv":

```
states,percent
UP,80.00
```

```
Maharastra,70.00
Bihar,65.0
MP,60.00
Gujarat,50.0
WB,49.0
TN,35.0
```

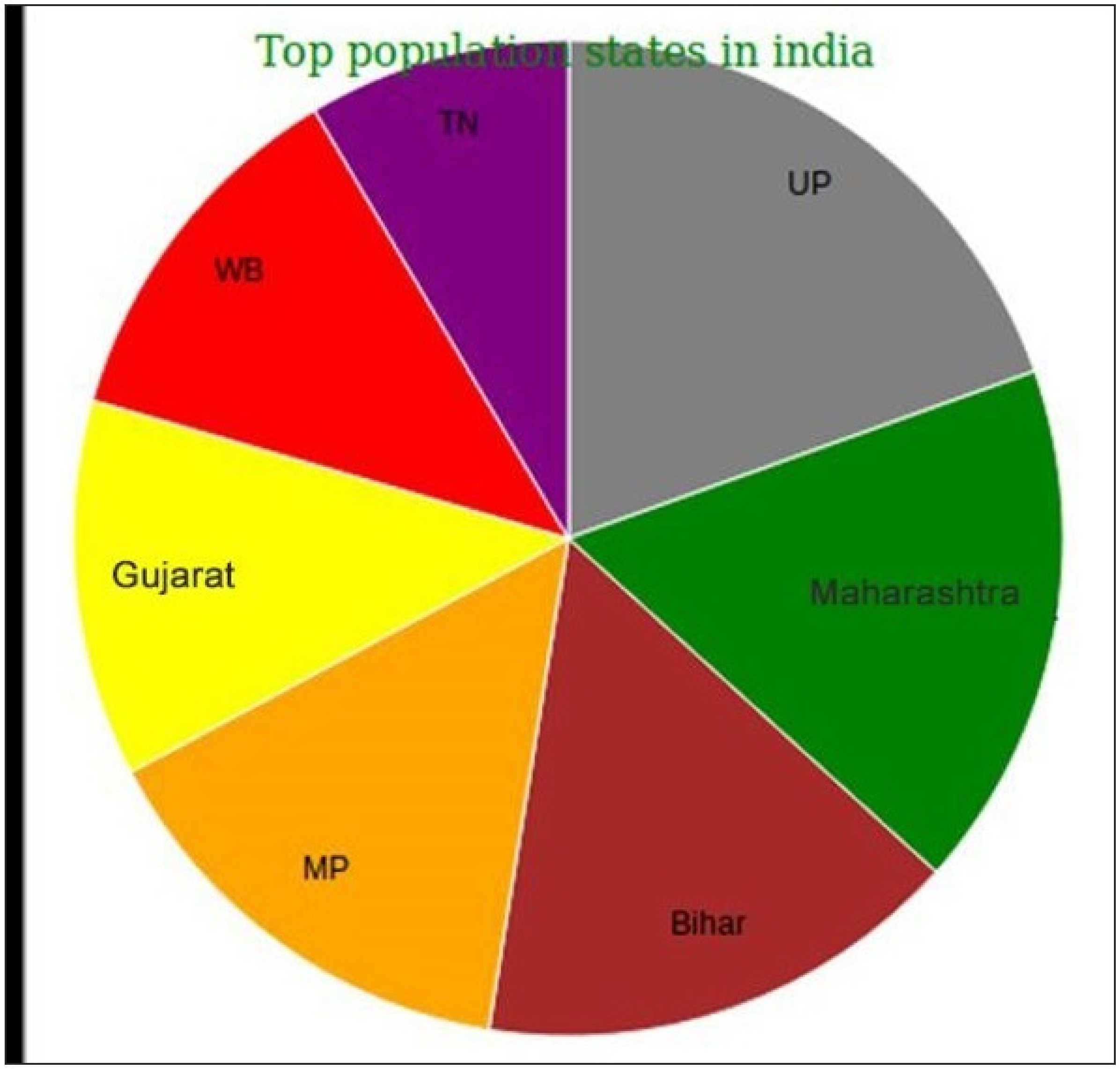
Let us create a pie chart visualization for the above dataset. Create a webpage "piechart.html" and add the following code in it.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         .arc text {
            font: 12px arial;
            text-anchor: middle;
         }

         .arc path {
            stroke: #fff;
         }

        .title {
            fill: green;
            font-weight: italic;
         }
      </style>

      <script src = "https://d3js.org/d3.v4.min.js"></script>
   </head>
```

```
<body>
   <svg width = "400" height = "400"></svg>
   <script>
      var svg = d3.select("svg"),
         width = svg.attr("width"),
         height = svg.attr("height"),
         radius = Math.min(width, height) / 2;

      var g = svg.append("g")
         .attr("transform",
               "translate(" + width / 2 + "," + height / 2 + ")");

      var color = d3.scaleOrdinal([
   'gray', 'green', 'brown', 'orange', 'yellow', 'red', 'purple'
      ]);

      var pie = d3.pie().value(function(d) {
         return d.percent;
      });

      var path = d3.arc()
         .outerRadius(radius - 10).innerRadius(0);

      var label = d3.arc()
         .outerRadius(radius).innerRadius(radius - 80);

      d3.csv("populations.csv", function(error, data) {
         if (error) {
            throw error;
         }

         var arc = g.selectAll(".arc")
            .data(pie(data))
            .enter()
            .append("g")
            .attr("class", "arc");
            arc.append("path")
               .attr("d", path)
               .attr("fill", function(d) {
                                return color(d.data.states);
                    });

            console.log(arc)

            arc.append("text").attr("transform", function(d) {
               return "translate(" + label.centroid(d) + ")";
            })

            .text(function(d) { return d.data.states; });
         });

         svg.append("g")
            .attr("transform",
                  "translate(" + (width / 2 - 120) + "," + 20 + ")")
            .append("text").text("Top population states in india")
            .attr("class", "title")
      </script>
   </body>
</html>
```

Output:


Top population states in india
TN
UP
WB
Gujarat
Maharashtra
MP
Bihar

7.5. Donut Chart

Donut or Doughnut chart is just a simple pie chart with a hole inside. We can define the hole radius to any size you need, both in percent or pixels. We can create a donut chart instead of a pie chart. Change the inner radius of the arc to use a value greater than zero. It is defined as follows.

Example:

```
var arc = d3.arc()
   .outerRadius(radius)
   .innerRadius(100);
```

Same as the pie chart coding and with a slightly changed inner radius, we can generate a donut chart. Create a webpage dounutchart.html and add the following changes in it.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         .arc text {
            font: 12px arial;
            text-anchor: middle;
         }

         .arc path {
            stroke: #fff;
         }

         .title {
            fill: green;
            font-weight: italic;
         }
      </style>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
   </head>

   <body>
      <svg width = "400" height = "400"></svg>
      <script>
         var svg = d3.select("svg"),
            width = svg.attr("width"),
            height = svg.attr("height"),
            radius = Math.min(width, height) / 2;

         var g = svg.append("g")
            .attr("transform",
                  "translate(" + width / 2 + "," + height / 2 + ")");

         var color = d3.scaleOrdinal([
         'gray', 'green', 'brown', 'orange', 'yellow', 'red', 'purple'
         ]);

         var pie = d3.pie().value(function(d) {
            return d.percent;
         });

         var path = d3.arc()
            .outerRadius(radius)
            .innerRadius(100);

         var label = d3.arc()
            .outerRadius(radius)
            .innerRadius(radius - 80);

         d3.csv("populations.csv", function(error, data) {
            if (error) {
               throw error;
            }
            var arc = g.selectAll(".arc")
               .data(pie(data))
               .enter()
               .append("g")
               .attr("class", "arc");
               arc.append("path")
                  .attr("d", path)
                  .attr("fill", function(d) { return
color(d.data.states); });

               console.log(arc)

               arc.append("text")
                  .attr("transform", function(d) {
                     return "translate(" + label.centroid(d) + ")";
                   })
                  .text(function(d) { return d.data.states; });
         });

         svg.append("g")
            .attr("transform", "translate(" + (width / 2 - 120) + "," +
20 + ")")
            .append("text")
            .attr("class", "title")
      </script>
   </body>
</html>
```

Here, we have changed the path variable as –

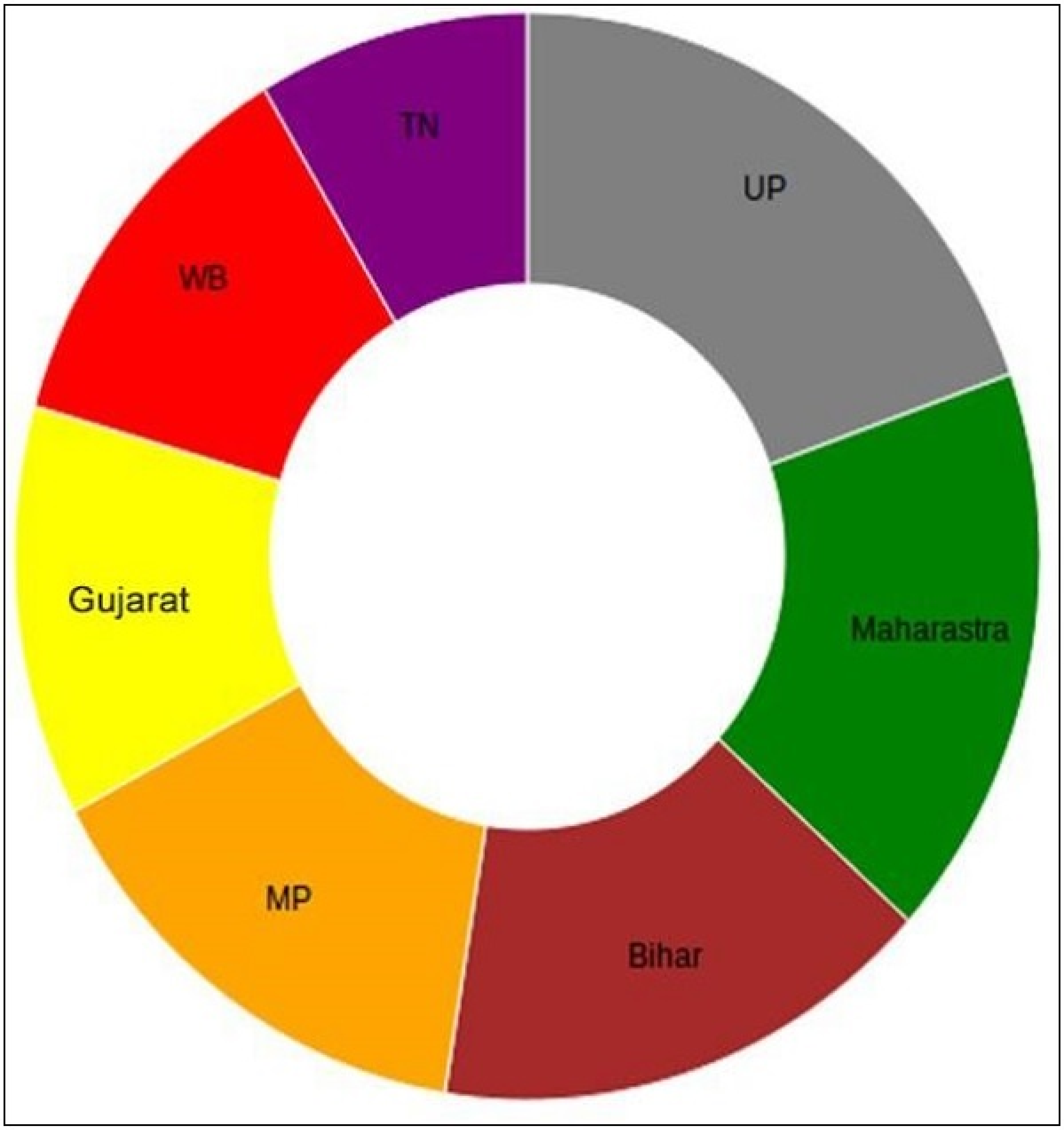
Example:

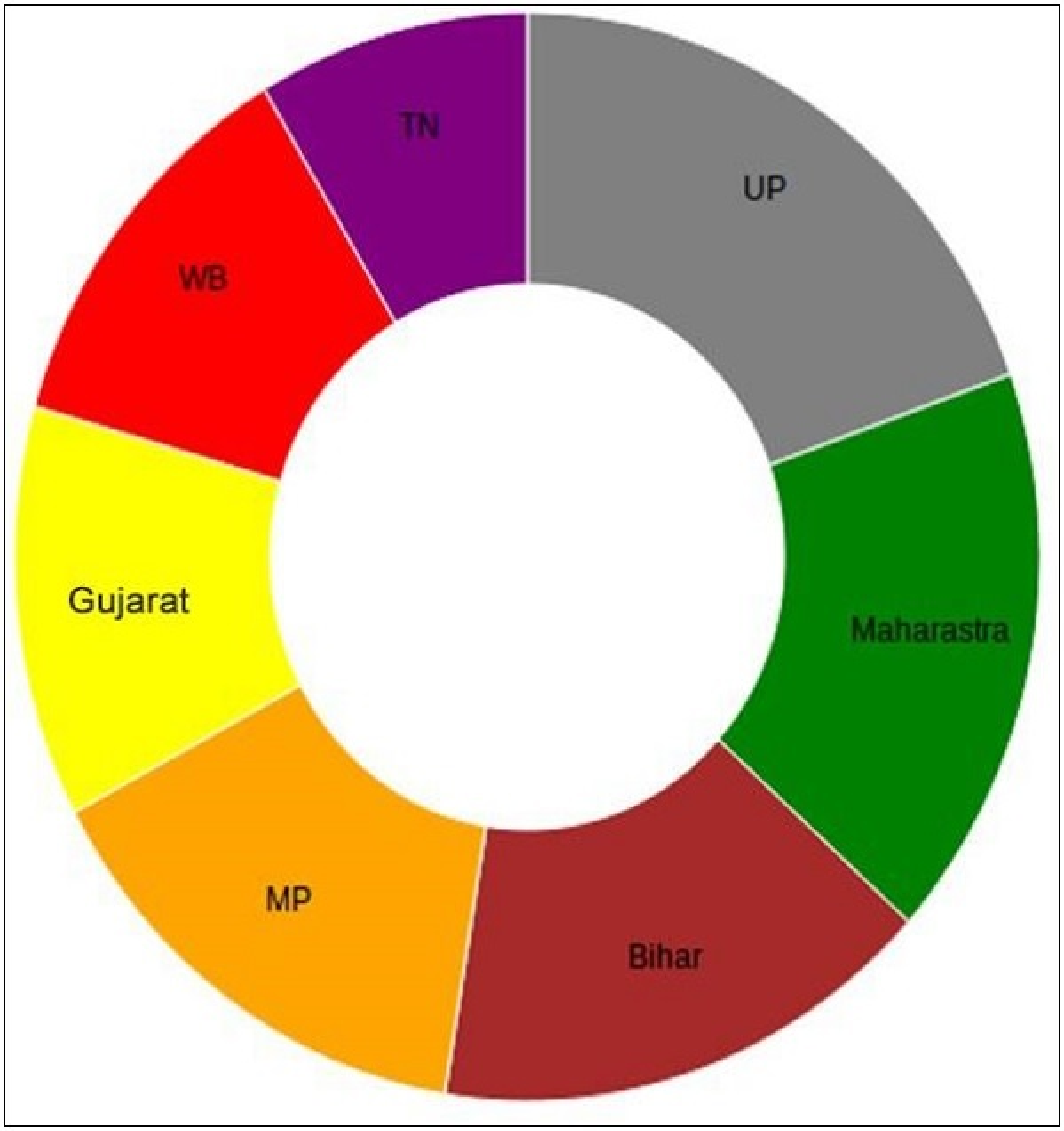
```
var path = d3.arc()
   .outerRadius(radius)
```

```
        .innerRadius(100);
```

We set the innerRadius value>0 to generate a donut chart. Now, request the browser and we can see the following response.

Output:




TN
UP
WB
Gujarat
Maharastra
MP
Bihar