CMPT 384 – Information Visualization

D3.js

Lab 2

Course Instructor: Debajyoti Mondal

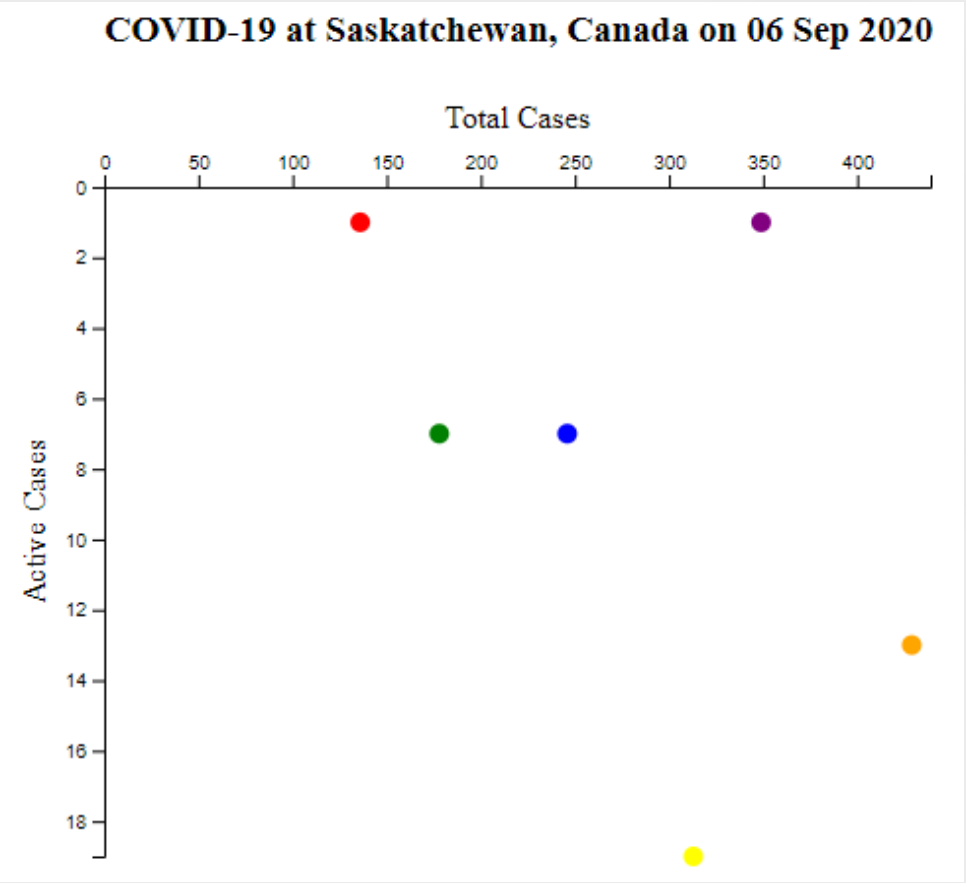
Lab Tutorial Instructor : **Arman Heydari**(arman.heydari@usask.ca)

Agenda

- D3 default max, min function
- D3 Scales – Linear scale
- D3 Axis (left and top)
- D3 Transformation
- D3 scatter plot draw*

cases_6Sep20.csv

```
Region,Total_Cases,Recovered,Deaths,Color
Far North,349,341,7,purple
North,246,233,6,blue
Central (excluding Saskatoon),178,169,2,green
Saskatoon,313,292,2,yellow
South (excluding Regina),429,410,6,orange
Regina,136,134,1,red
```



*no legend and no tooltip

D3 Min and Max

- These are default functions of D3 library

For an array of Int

```
> d3.max([2, 5, 7, 1, 3]);
<· 7
> d3.min([2, 5, 7, 1, 3]);
<· 1
```

```
> var array = [2, 4, 5, 1];
<· undefined
> d3.max(array);
<· 5
```

For an array of object

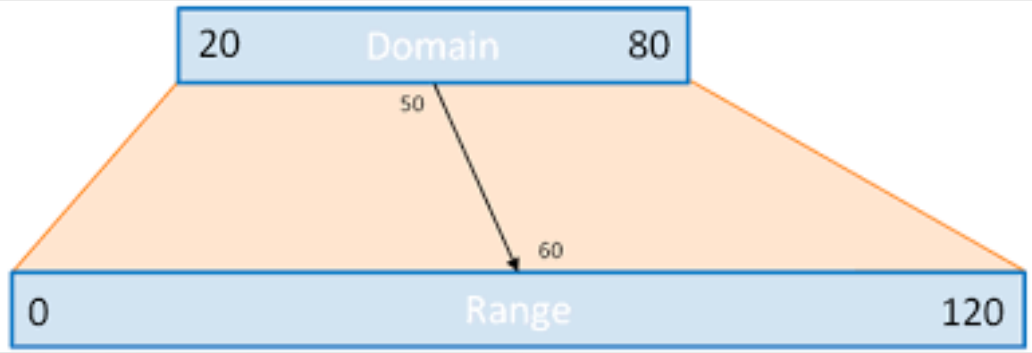
```
flights = [
  {airline: "Icelandair", price: 1621, stops: 3},
  {airline: "Multiple airlines", price: 1381, stops: 2},
  {airline: "Air France", price: 1948, stops: 0},
  {airline: "WestJet", price: 1711, stops: 1},
  {airline: "Air France", price: 1951, stops: 1},
  {airline: "French Bee", price: 1780, stops: 1}
];
```

```
> d3.min(flights, d=>d.stops);
<· 0
> d3.max(flights, d=>d.price);
<· 1951
> d3.min(flights, d=>d.price);
<· 1381
```

D3 Scales

- **Continuous Scale**
  - **Linear**, Power, Log, Time
- Sequential Scale
- Diverging Scale
- **Quantize Scale**
- Quantile Scale
- Ordinal Scale

D3 Scales – Linear Scale



http://www.jeromecukier.net

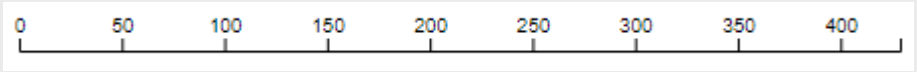
```
var xScale = d3.scaleLinear()
    .domain([20, 80])
    .range([0, 120]);
```

```
> xScale(20)
<- 0
> xScale(80)
<- 120
> xScale(50)
<- 60
```

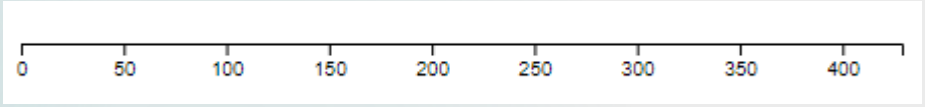
Domain – what we have, e.g. data range
Range – what we want, e.g. pixel range for chart

D3 Axis

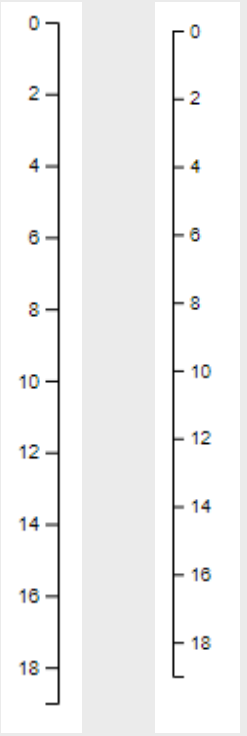
- **Top Axis : d3.axisTop(<need to pass a scale>)**



- **Bottom Axis : d3.axisBottom(<need to pass a scale>)**



- **Left Axis : d3.axisLeft(<need to pass a scale>)**
- **Right Axis : d3.axisRight(<need to pass a scale>)**



Left

Right

D3 - Transformation

<text>hello</text>

- **Translation:**

<text transform="translate(10, 10)"></text>

- **Rotation:**

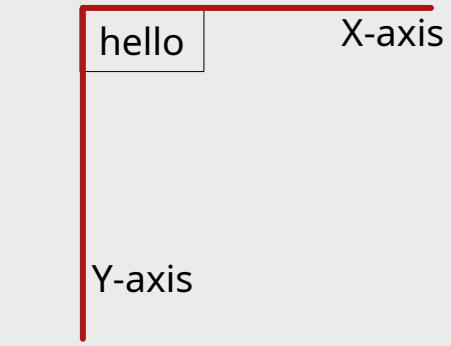
<text transform="rotate(-45)"></text>

<text transform="rotate(45)"></text>

- **Translation and Rotation:**

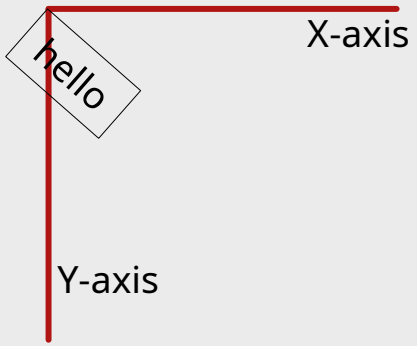
<text transform="translate(10, 10) rotate(45)"></text>

- **Scale, Skew**

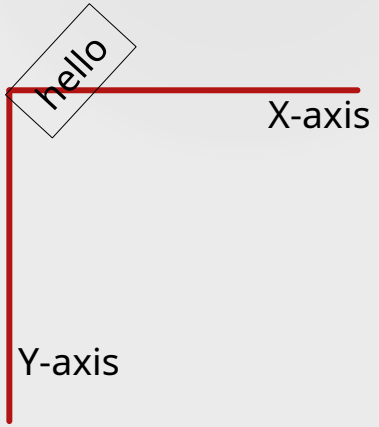


before any transformation

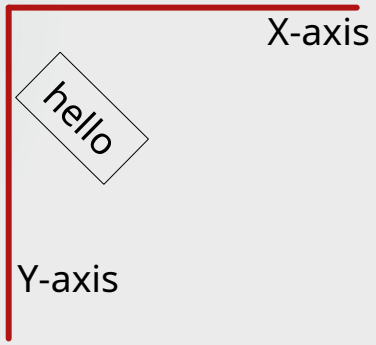
translate(10, 10)



rotate(45) - clockwise

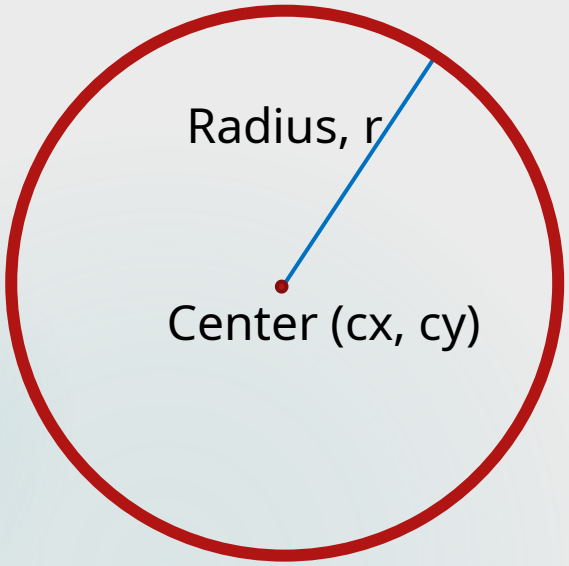


rotate(-45) – anti-clockwise



translate(10, 10) rotate(45)

D3 Circle draw

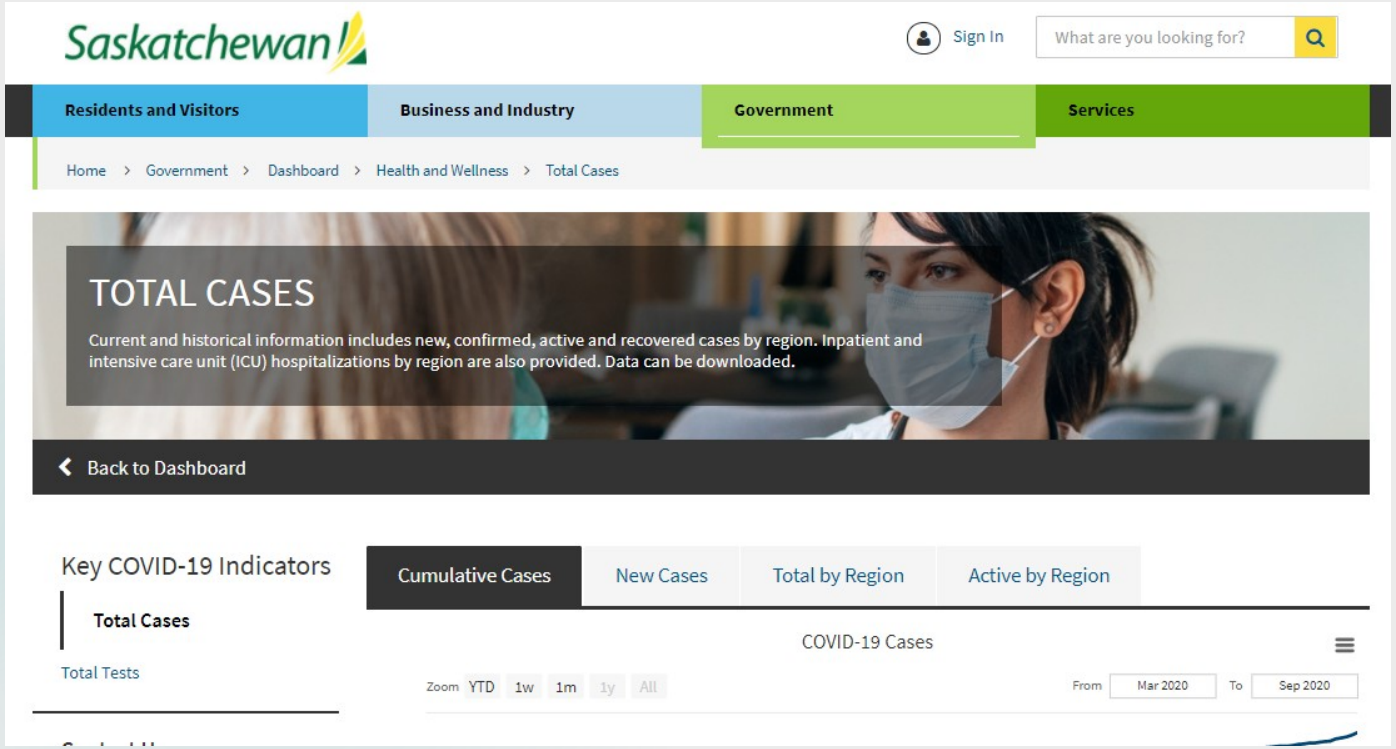


<circle r="5" cx=“10" cy=“8“></circle>

- Fill : define the color to fill the circle
- Stroke : define the color of the border
- Stroke-width : define the width of the border

Data Source

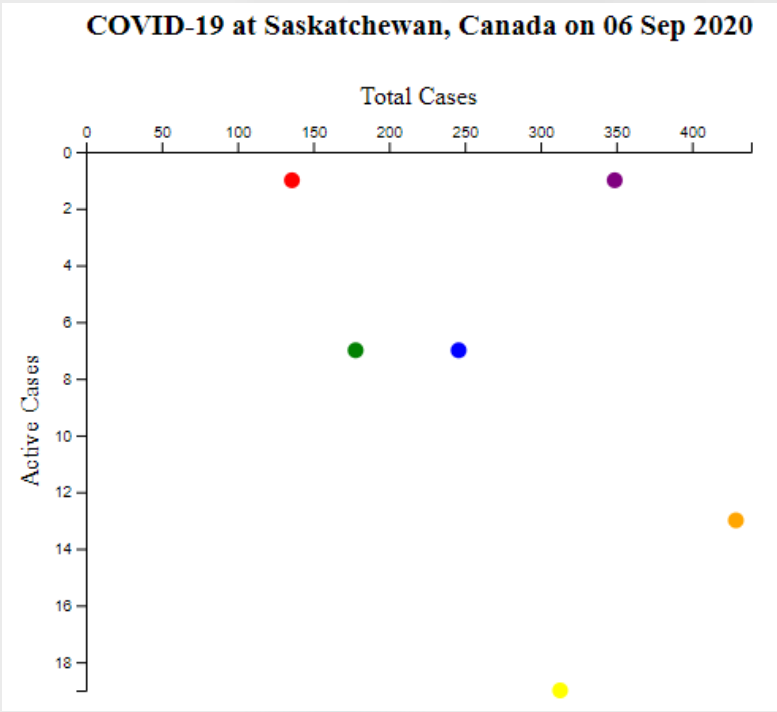
COVID-19 Cases at Saskatchewan, Canada on 06 September 2020



https://dashboard.saskatchewan.ca/health-wellness/covid-19/cases

cases_6Sep20.csv

```
Region,Total_Cases,Recovered,Deaths,Color
Far North,349,341,7,purple
North,246,233,6,blue
Central (excluding Saskatoon),178,169,2,green
Saskatoon,313,292,2,yellow
South (excluding Regina),429,410,6,orange
Regina,136,134,1,red
```



Start the server

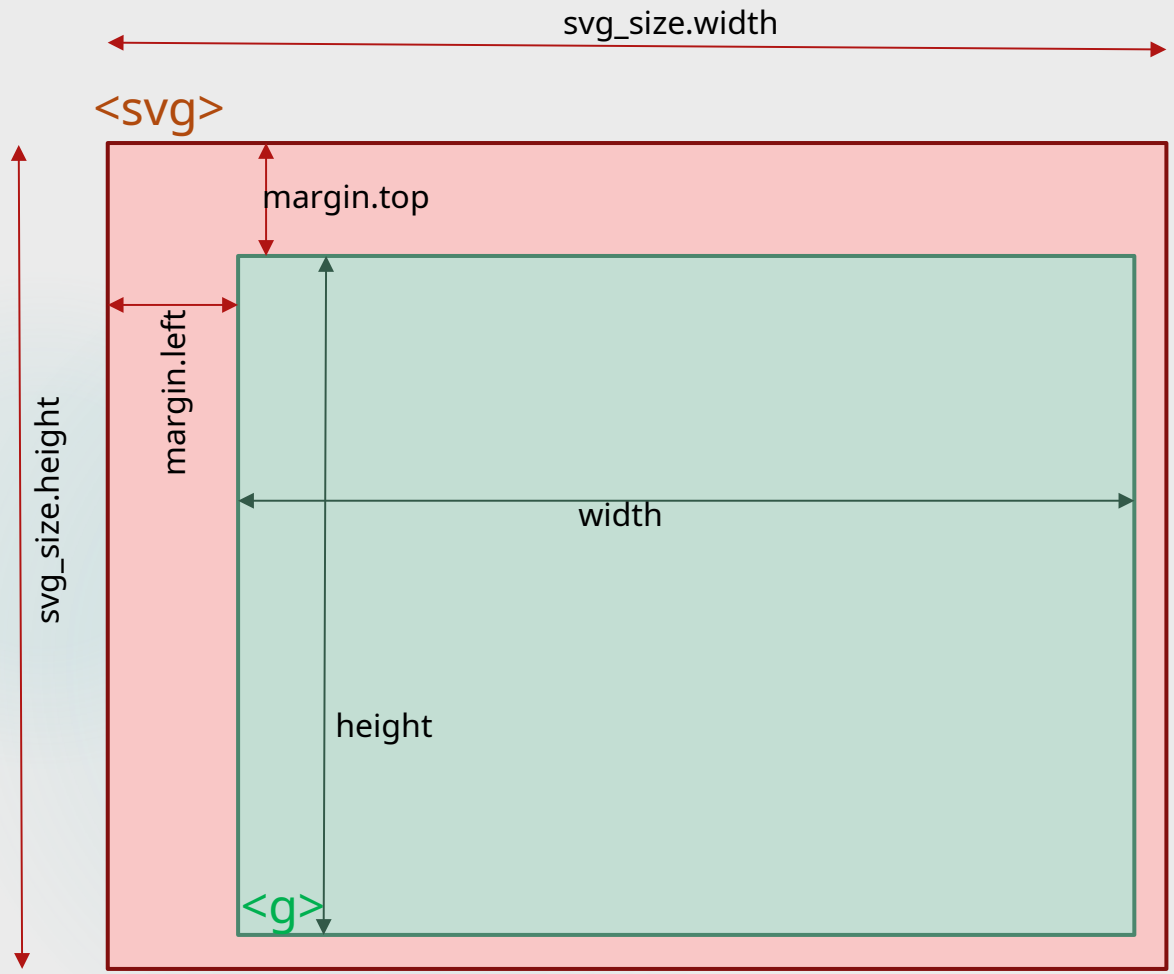
Go to your folder and run the server
First command is for python version < 3.0 and the second one if python version >= 3.0.

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\> python -m SimpleHTTPServer 8888
C:\Users\jyoti\AppData\Local\Programs\Python\Python37-32\python.exe: No
module named SimpleHTTPServer

C:\> python -m http.server 8888
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
```

Scatter plot - margin




```
// This is margin for the chart. It is required since we have axis labels.
// Axis labels required some spaces (left and top).
var margin = {top: 50, right: 10, bottom: 10, left: 50}
// Size for the SVG element
var svg_size = {width: 500, height: 400};

// Actual available width and height for the chart have been calculated after subtracting the margin
var width = svg_size.width - margin.left - margin.right;
var height = svg_size.height - margin.top - margin.bottom;
```

Scatter plot – Read CSV

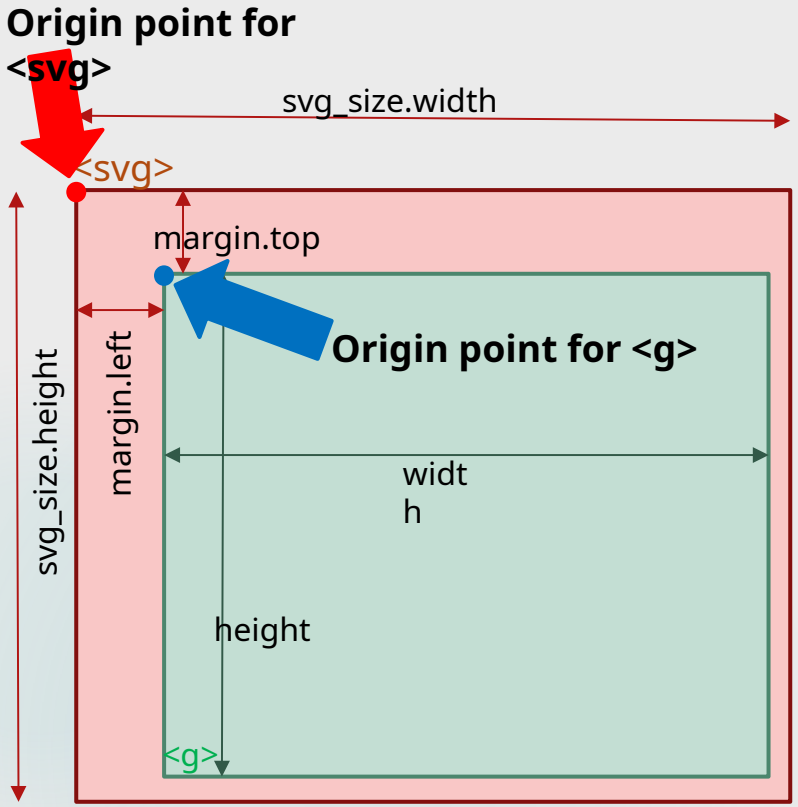
- **Good practice to do all data pre-processing, modification, cleaning or filling before creating the chart**

```
d3.csv("cases_6Sep20.csv").then(function(data){
    // You should do data pre-processing or modification or cleaning or filling in here
    // before creating the chart

    // Active cases have been calculated
    // Both for loop does the same work. Follow whatever you like
    data.forEach(function(d){
        d.Active_Cases = d.Total_Cases - d.Recovered - d.Deaths;
    });
    //for(var i=0; i< data.length; i++)
    //{
    //    data[i].Active_Cases = data[i].Total_Cases - data[i].Recovered - data[i].Deaths;
    //}
    console.log(data);
    generateVisualization(data);
});
```

Active Cases = Total Cases – Recovered Cases - Deaths

Scatter plot – svg_g variable



- **append <svg> tag and set width and height**
- **append <g> tag and translate the origin into blue dot**
- **<g> is stored in svg_g variable. Whatever tag we add inside this <g>, it will count the origin as this blue dot (not the red dot)**

```
// svg element has appended into div with id='my_chart'
// svg width and height has been set also
var svg_g = d3.select('#my_chart')
              .append('svg')
              .attr('width', svg_size.width)
              .attr('height', svg_size.height)
              // Inside svg a graphic element 'g' tag has been added and it has been translated to
              // the top-left (inner side) of the margin
              .append('g')
              .attr('transform', 'translate('+ margin.left +', '+ margin.top +')');
```

Scatter plot – X axis

- Calculate the maximum Total Cases and Active cases

```
// Maximum value have been calculated for Total Cases and Active Cases
max_active_case = d3.max(dataset, d => d.Active_Cases);
max_total_case = d3.max(dataset, d => d.Total_Cases);
```

- X axis scale is defined

```
// X axis scale has been defined
// domain is the range of our data (what we have)
// range is the range of available pixels for the chart (what we want)
var x_axis_scale = d3.scaleLinear()
                     .domain([0, parseInt(max_total_case)])
                     .range([0, width]);
```

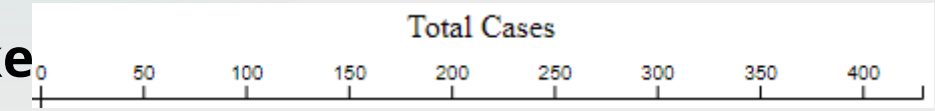
- X axis has been created

```
// X axis has been created at the top
var x_axis = svg_g.append('g')
                  .call(d3.axisTop(x_axis_scale));
```

- X axis label has been placed

```
// X axis label has been created
// It has been translated at the middle of the X axis
var x_axis_label = svg_g.append('text')
                        .attr("transform", 'translate(' + width/2 +', -30)')
                        .style('text-anchor', 'middle')
                        .text('Total Cases');
```

Looks like this :



Text-anchor Style

text-anchor = middle

Total Cases

text-anchor = start

Total Cases

text-anchor = end

Total Cases

- Position (x, y) co-ordinate of the text element

Scatter plot – Y axis

- Y axis scale is defined

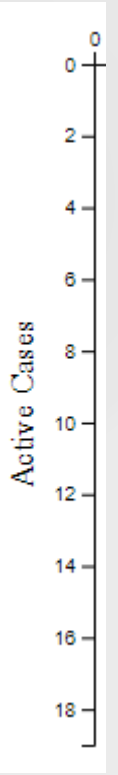
```
// Y axis scale has been defined
// domain is the range of our data (what we have)
// range is the range of available pixels for the chart (what we want)
var y_axis_scale = d3.scaleLinear()
                     .domain([0, max_active_case])
                     .range([0, height]);
```

- Y axis label has been placed

```
// Y axis label has been created
// It has been translated at the middle of the Y axis and then rotated 90 degree (anti-clockwise)
var y_axis_label = svg_g.append('text')
                        .attr("transform", 'translate(-30, '+ height/2 +') rotate(-90)')
                        .style('text-anchor', 'middle')
                        .text('Active Cases');
```

- Translate the label into the middle of y axis by **translate(-30, height/2)** and rotate it 90 degree anti-clockwise by **rotate(-90)**

Looks like this :

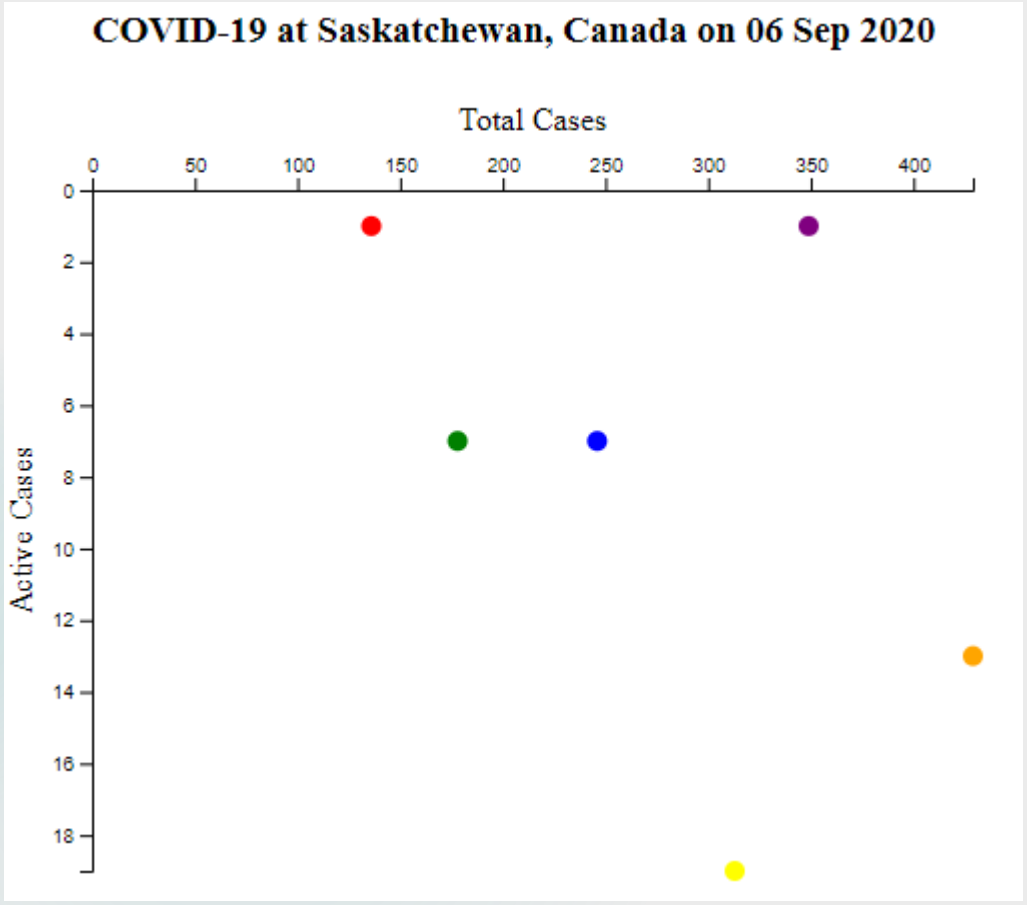


Scatter plot – drawing circles

```
// Creating all the circles
var circles = svg_g.append('g')
                    // Creating virtual array since circle does not exist in g element
                    .selectAll('circle')
                    // Join data with virtual array
                    .data(dataset)
                    .enter()
                    // From this point, everything will be iterated
                    // Appending circles
                    .append('circle')
                    .attr('r', 5)
                    .attr('cx', function(d){
                        // Caliing x_axis_scale for values
                        return x_axis_scale(d.Total_Cases);
                    })
                    .attr('cy', function(d){
                        // Caliing y_axis_scale for values
                        return y_axis_scale(d.Active_Cases);
                    })
                    .attr('fill', function(d){
                        return d.Color;
                    })
                    .attr('title', function(d){
                        return d.Region;
                    });
```

x_axis_scale and y_axis_scale have been used to get the actual cx and cy (pixel) values for the chart

Scatter plot - finish



Those who love challenges – try to draw scatter plot with bottom(x-axis) and right (y-axis)

N.B. – To do that you might need to translate the axis at bottom or to the right