Assignment 4: - GIT(hub) Viz

Code Repository URLs:
https://github.com/qroll/CS3219-Assignment4

Name	See Loo Jane	Teo Yi Rong	Quek Ruo Ling
Matriculation Number	A0131003J	A0121814R	A0131507R

1. Introduction

Through this assignment, we were able to understand the importance of Data Visualization as well as a powerful tool, d3, to create them. On top of this, we also learnt how to extract the data we require from JSON that github returns using Python code. This

Ruo Ling implemented the first two visualisations, Jane implemented the third visualization and Yi Rong was in charge of the documentation.

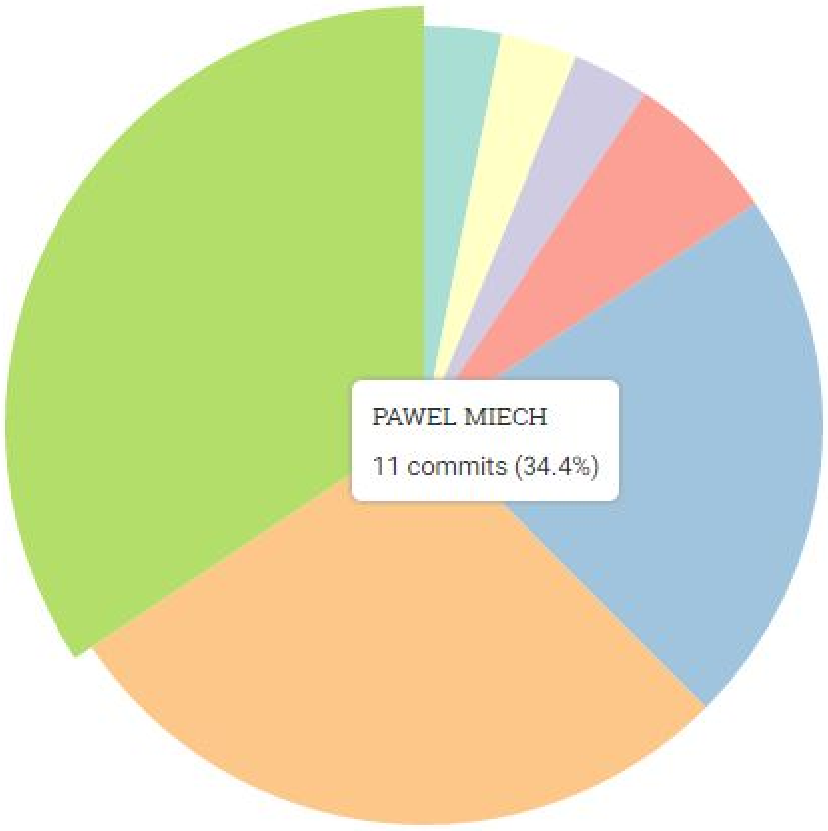
2. Visualizations

Objective	**Visualization**
1	Pie Chart
2	Stacked Bar Chart
3	Stacked Bar Chart

1. Commits by each author (as a percentage of the total commits)

As we need to show the data as a percentage of the total, we decided to visualize the information using a Pie Chart. This allows the viewer to easily discern how much each author did in comparison to the total as well as to their peers.

For this implementation, we decide to hide the names of the authors and only display their name as well as their commit count as a tooltip on mouseover. This was because we found the Pie Chart to be extremely cluttered if we displayed this information all at once.



Step 1: Retrieving the data

We obtained the number of commits made by each author, between 1st June 2016 and 1st July 2016, using the git command:

```
git log --pretty=format:%aN --since="2016-06-01 00:00:00"
--until="2016-07-02 00:00:00" | sort
```

What we got was :

```
Elias Dorneles 1
```

```
Joakim Uddholm 7
Matt O'Connell 1
Matvei Nazaruk 1
Mikhail Korobov 2
Paul Tremberth 9
Pawel Miech 11
```

Step 2: Javascript array

We inserted this data into an array in our js script like so:

```
var dataset = [
      { label: 'Elias Dorneles', count: 1 },
      { label: 'Matt O\'Connell', count: 1 },
      { label: 'Matvei Nazaruk', count: 1 },
      { label: 'Mikhail Korobov', count: 2 },
      { label: 'Joakim Uddholm', count: 7 },
      { label: 'Paul Tremberth', count: 9 },
      { label: 'Pawel Miech', count: 11 }
];
```

Step 3: Creating the visualization

We use the arc function to define the radius of the pie chart.

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

The data is input into the d3 pie function, which computes the size of each pie segment.

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

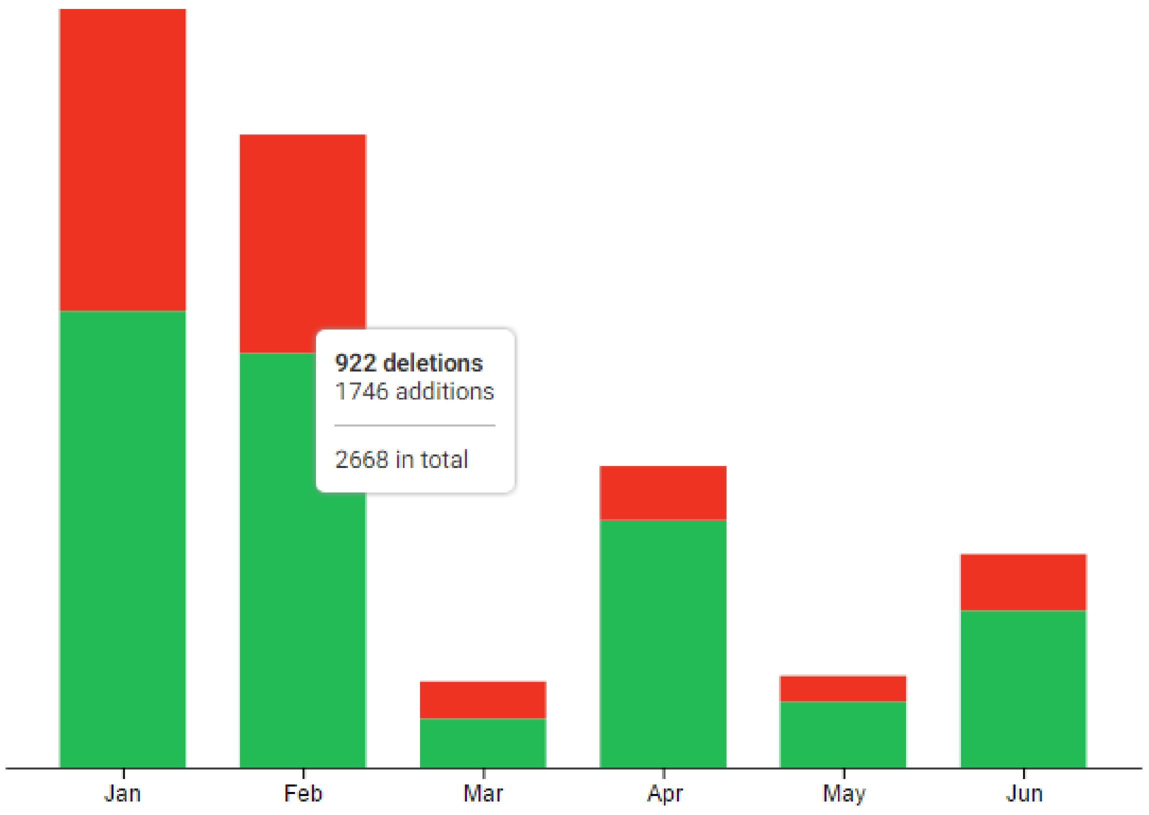
Finally, the following code performs the actual creation of elements in the svg.

```
var path = svg.selectAll('path')
  .data(pie(dataset)).enter().append('path')
  .attr('d', arc)
  .attr('fill', function(d) {
      return color(d.data.label);
  });
```

We also pull information in the tooltip from the data. By setting the event listeners for mouseover and mouseout, the tooltip appears and disappears accordingly when the mouse hovers over the pie.

2. Sums of additions by all authors and sums of deletions by all authors

As this visualization requires the display of two variables, we elected to use a bar chart. To make it easier to compare between the number of additions and deletions, we used a stacked bar chart to highlight the difference. Similar to the Pie Chart, instead of displaying data constantly, we chose to show detailed data only upon mouseover of the specific bar. As an added touch we bolded the font of the additions or deletions tooltip depending on whether the additions or the deletions part of the bar is moused over.



Step 1: Retrieving the data

Firstly, we required the number of additions and deletions made by each author between 1st June 2016 and 1st July 2016. We got this data using the git-command:

```
git log --since="2016-06-01 00:00:00"  --before="2016-07-01 00:00:00"
--numstat --pretty=format:"" | awk -F '[[:space:]a-zA-Z]+' '{add+=$1;
del+=$2}END{print add, del}'
```

What we got was :

```
01 Jan - 31st Jan: 1924a 1272d
01 Feb - 29 Feb: 1746a 922d
```

```
01 Mar - 31Mar: 205a 157d
01 Apr - 30Apr: 1041a 229d
01 May - 31May: 278a 108d
01 June - 30June: 661a 238d
```

Step 2: Javascript array

We then inserted this data into an array in our js script like so:

```
var dataset = [
      { label: 'Jan', additions: 1924, deletions: 1272 },
      { label: 'Feb', additions: 1746, deletions: 922 },
      { label: 'Mar', additions: 205, deletions: 157 },
      { label: 'Apr', additions: 1041, deletions: 229 },
      { label: 'May', additions: 278, deletions: 108 },
      { label: 'Jun', additions: 661, deletions: 238 },
];
```

Step 3: Creating the visualization

Using this data, we were able to fill the y axis using:

```
var y = d3.scaleLinear()
      .domain([0, d3.max(dataset.map(function(d) {return d.additions +
d.deletions;}))])
      .range([0, height]);
```

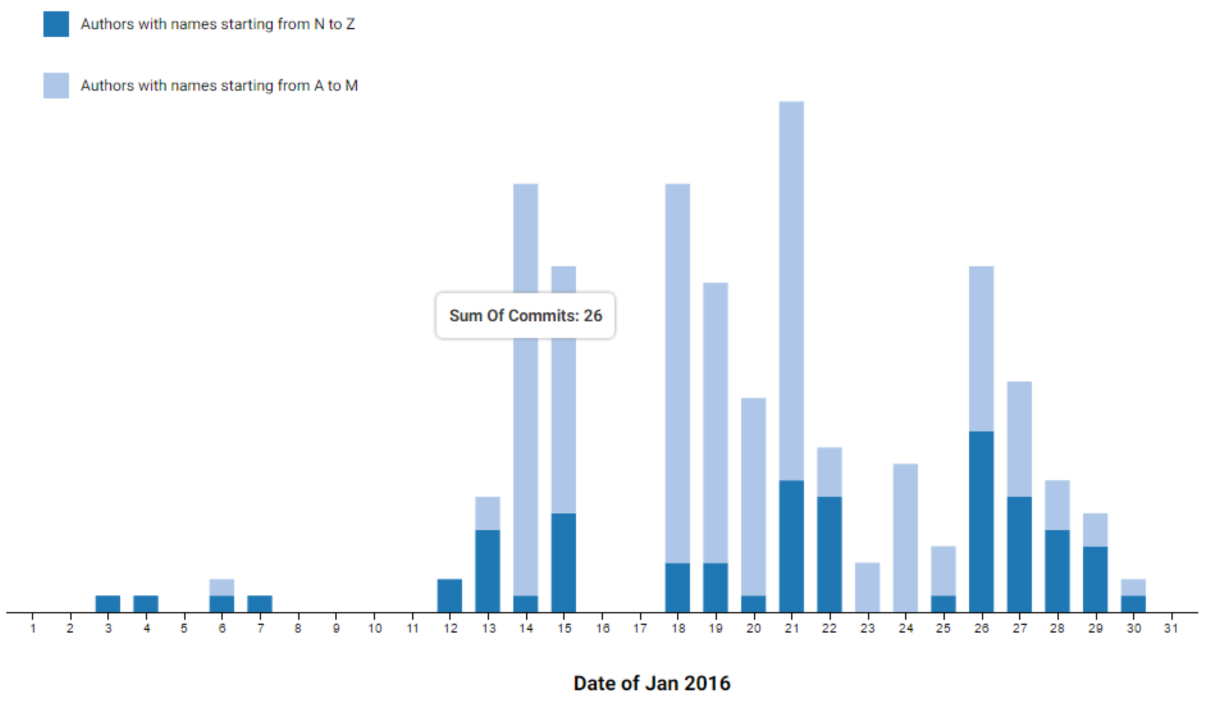
and the x axis using:

```
var x = d3.scaleBand()
            .range([0, width])
            .padding(0.3)
            .domain(dataset.map(function(d) {return d.label}));
```

The data is then used to populate the bar chart.

3. Sum of commits by the two groups of programmers for each day

As this visualization requires the display of two variables, we elected to use a bar chart. To make it easier to compare between the number commits between two groups, we elected to go for a stacked bar chart to highlight the difference. As this chart is a lot bigger and we have a lot more white space to work with, we decided to place a legend in the top left corner of the graph. Also, we added a tooltip which displays the exact number of commits for each stacked bar chart. This makes the data easier to interpret at a glance.



Step 1: Retrieving the data

Firstly, we acquired the number of commits made by each group of authors between 1st Jan 2016 and 1st Feb 2016. The output is formatted in the form of “<commit-date> <author name>”, and then sorted according to the date. We got this data using the git-command:

```
git log --pretty=format:’%cI %aN’ --since="2016-01-01 00:00:00"
--until="2016-02-01 00:00:00" | sort
```

As the resulting list is too long, we decide to not to include it here.

Step 2: Javascript array

We then used python script to format the output into an array of JSON objects, where “date” represents the numerical date of January 2016, “n-z” represents the sum of commits made by authors of name starting from n to z, “a-m” represents the sum of commits made by authors of name starting from a to m. A snippet of the final output is as follow:

```
var data = [
    {'date': 1, 'n-z': 0, 'a-m': 0},
    {'date': 2, 'n-z': 0, 'a-m': 0},
    {'date': 3, 'n-z': 1, 'a-m': 0},
    {'date': 4, 'n-z': 1, 'a-m': 0},
    {'date': 5, 'n-z': 0, 'a-m': 0},
    {'date': 6, 'n-z': 1, 'a-m': 1},
    ...
];
```

Step 3: Creating the visualization

Next, we add an svg element to the HTML placeholder called "chart3":

```
var svg = d3.select("#chart3")
        .append("svg")
        .attr("width", width + margin.left + margin.right)
        .attr("height", height + margin.top + margin.bottom)
        .append("g")
        .attr("transform", "translate(" + margin.left + "," + margin.top
+ ")");
```

To use d3 stack() to prepare a data set that can be used for stacked graphs, we pump our original data into the stack() function and use keys() to tell d3 what each category is:

```
var dataStackLayout = d3.stack()
                 .keys(keys)
                 .order(d3.stackOrderNone)
                 .offset(d3.stackOffsetNone);
```

This data can now be used for the “.stack” component of the svg we previously created, which we call “layer”:

```
var layer = svg.selectAll(".stack")
        .data(dataStackLayout(data))
        .enter().append("g")
        .attr("class", "stack")
        .style("fill", function (d, i) {
            return color(i);
        });
```

Lastly, we have to render the actual rectangles that we want to be stacked. The code below does it:

```
// Render the Rectangles + Attach tooltips
layer.selectAll("rect")
        .data(function (d) {
            return d;
        })
        .enter().append("rect")
        .attr("x", function (d) {
            return x(d.x);
        })
        .attr("y", function (d) {
            return y(d.y + d.y0);
        })
        .attr("height", function (d) {
            return y(d.y0) - y(d.y + d.y0);
        })
        .attr("width", x.rangeBand());
```

After getting the rectangles, we finally render the x-axis:

```
svg.append("g")
        .attr("class", "xAxisQ3")
        .attr("transform", "translate(0," + height + ")")
        .call(xAxis);
```

For this visualisation, we only chose to display the x-axis and not the y-axis because 1) the tooltips can show the exact number of commits for each bar chart; 2) the y-axis will add clutter to the graph without value adding the data respresentation, as the height of the bars already serve as a very clear indication of the proportion of commits.

Finally, we get a stacked bar chart where the highest total number of commits made by both groups is the max of the y-axis and the dates are the x-axis.

The subsequent code involves adding the tooltips (explanation omitted due to similarity with that of Visualisation 1 and 2) and the legend. To create the legend, we first have to supply the data set of colors, after which we define the size and position of the legend as shown below:

```
var legend = svg.selectAll('.legend')
                .data(color.domain())
                .enter()
                .append('g')
                .attr('class', 'legend')
                .attr('transform', function(d,i) {
                    var height = legendRectSize + legendSpacing;
                    var offset = height * color.domain().length/2;
                    var horz = -2 * legendRectSize;
                    var vert = (i * height - offset) + 30;
                    return 'translate(' + 30 + ',' + vert + ')';
                });
```

Next, we just have to append the CSS elements that we want to be displayed in each item of the legend. In our case, we have a “rect” element and a “text” element for each legend item:

```
legend.append('rect')
        .attr('width', legendRectSize)
        .attr('height', legendRectSize)
        .style('fill', color)
        .style('stroke', color);

legend.append('text')
        .attr('x', legendRectSize + legendSpacing - 20)
        .attr('y', legendRectSize - legendSpacing + 25)
        .text(function(d) {
            if (d == 1) {
                return "Authors with names starting from A to M";
            } else {
                return "Authors with names starting from N to Z";
            }
        });
```