Mini-Project: GIT(Hub) Viz

Student Name	Zhang Mingmin	Tan Cher Wah
Matriculation Number	A0159339R	A0148575R

1. Introduction

In this assignment we provide 3 visualizations for github repo with visualization tools 'c3.js' and 'echarts.js'. A python script is written to grab and process the data, and an HTML page is written to show the visualizations. The default data set we used is from 'torvalds/linux', but other repos are also supported by passing user/repo as parameters to the python script. Zhang Mingmin handles the repo data collection and processing, as well as the first viz. Tan Cher Wah is responsible for the rest two viz.

2. Visualizations - Purpose & Method

(i) The visualizations we choose for each objective:

Objective	Visualization
1	Stacked Area Chart
2	Line Chart
3	Nested Pie Chart

(ii) For each of visualizations :

Objective 1:

Step1: use python module 'requests' to grab the json data from GitHub Api 'https://api.github.com/repos/torvalds/linux/stats/participation'

```
data = requests.get(
       "https://api.github.com/repos/torvalds/linux/stats/participation").json()
```

Step2: The data contains latest 52 weeks commit count by repo owner and all users, subtract the owner count from the all count to get count for others, save the data to a csv file with the order reversed

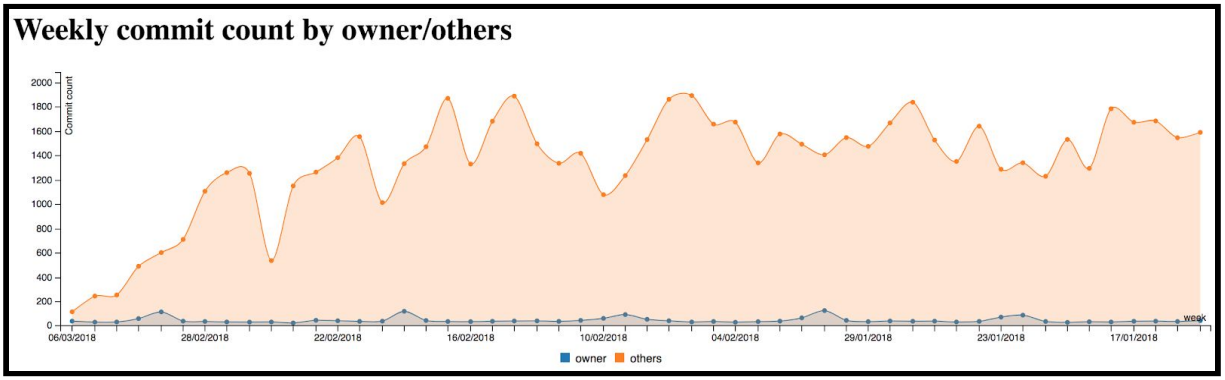
```
result = [data["owner"], [x-y for x, y in zip(data["all"], data["owner"])]]
with open('participation.csv', 'w') as file:
    wr = csv.writer(file, lineterminator='\n')
    wr.writerow(['owner', 'others'])
    for x in reversed(range(0, len(result[0]))):
        wr.writerow([result[0][x], result[1][x]])
```

Step3: Write c3.js script to read from the csv file and generate the chart

```
c3.generate({
          bindto: '#chart1',
          data: {
```

```
            url: 'participation.csv',
            types: {
                owner: 'area-spline',
                others: 'area-spline'
            },
            groups: [
                [
                    ['owner', 'others']
                ]
            ]
        },
        axis: {
            x: {
                tick: {
                    format: function (x) {
                        var today = new Date();
                        today.setDate(today.getDate() - x - 1);
                        return today.toLocaleDateString();
                    }
                },
                label: 'week'
            },
            y:{
                label: 'Commit count'
            }
        }
    });
```

result:



Objective 2:

Step1: use python module ‘requests’ to grab the json data from GitHub Api
‘https://api.github.com/repos/torvalds/linux/stats/punch_card‘

```
data = requests.get(
       "https://api.github.com/repos/torvalds/linux/stats/punch_card").json()
```

Step2: save the json data to csv file

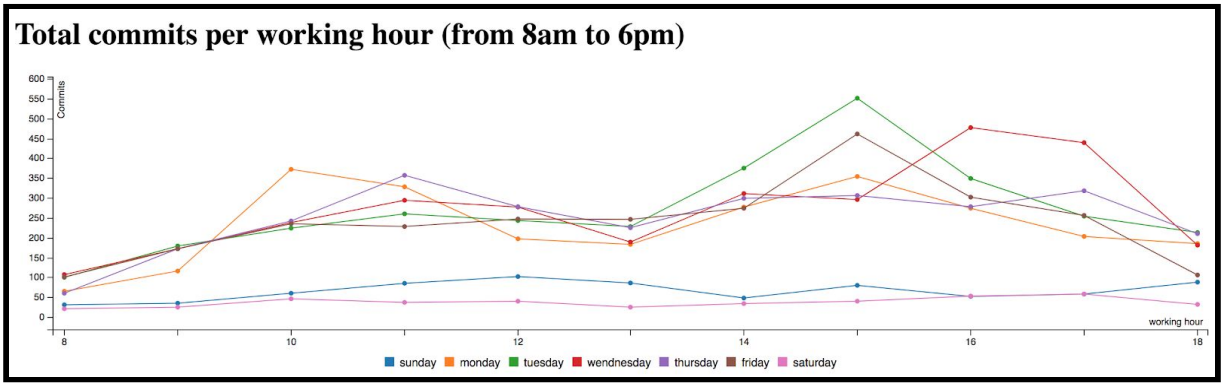
```
with open('workinghour.csv', 'w') as file:
   wr = csv.writer(file, lineterminator='\n')
   wr.writerow(['sunday', 'monday', 'tuesday', 'wendnesday',
                'thursday', 'friday', 'saturday'])
   for x in range(8, 19):
```

```
        hour = []
        for y in [a for a in data if a[1] == x]:
            hour.append(y[2])
        wr.writerow(hour)
```

Step3: Write c3.js script to draw the chart

```
c3.generate({
            bindto: '#chart2',
            data: {
                url: 'workinghour.csv',
                type: 'line'
            },
            axis: {
                x: {
                    tick: {
                        format: function (x) {
                            return x + 8;
                        }
                    },
                    label: 'working hour'
                },
                y:{
                    label: 'Commits'
                }
            }
        });
```

result:



Objective 3:

Step1: use python module ‘requests’ to grab the json data from GitHub Api ‘https://api.github.com/users/torvalds/repos’ to get all the repos of the user, for each of the repo, use GitHub Api ‘https://api.github.com/repos/torvalds/" + repos[index]["name"] + "/languages’ to get the languages of each repo, at the same time merge all the returned json data into one, save to a json file for language ‘C’ and others, as well as another json file for all the other languages

```
repos = requests.get("https://api.github.com/users/torvalds/repos").json()
def mergeToLanguagesBytes(input):
   for propertyName in input:
        if propertyName in languagesBytesAll:
            languagesBytesAll[propertyName] += input[propertyName]
        else:
            languagesBytesAll[propertyName] = 0
            languagesBytesAll[propertyName] += input[propertyName]
```

```
def split():
   largest = max(languagesBytesAll.iteritems(), key=operator.itemgetter(1))[0]
   languagesBytesTopandOthers[largest] = languagesBytesAll[largest]
   for x in languagesBytesAll:
       if x != largest:
           languagesBytesTopandOthers["Others"] += languagesBytesAll[x]
           languagesBytesOthers[x] = languagesBytesAll[x]
def getLanguages(repos, index):
   if(index >= len(repos)):
       split()
       writeJsonToFile('languages.json', languagesBytesOthers)
       writeJsonToFile('languagesTopAndOthers.json',
                       languagesBytesTopandOthers)
       with open('languages.csv', 'w') as file:
           wr = csv.writer(file, lineterminator='\n')
           wr.writerow(['language', 'bytes'])
           for x in languagesBytesAll:
               wr.writerow([x, languagesBytesAll[x]])
       return
   if os.path.isfile('json/'+user+'_'+repo+'_'+repos[index]["name"]+'.json'):
       languages = json.load(
           open('json/'+user+'_'+repo+'_'+repos[index]["name"]+'.json'))
   else:
       languages = requests.get(
           "https://api.github.com/repos/"+user+"/" + repos[index]["name"] +
"/languages").json()
       writeJsonToFile('json/'+user+'_'+repo+'_' +
                       repos[index]["name"]+'.json', languages)
   mergeToLanguagesBytes(languages)
   getLanguages(repos, index+1)
   return
```

Step2: use jquery to get the json files and generate chart with echarts.js

```
$.getJSON('languages.json', function (json) {
               var legend = Object.keys(json)
               var arr = Object.keys(json).map(function (k) {
                   return {name: k,value: json[k]}
               }).sort(function (a, b) {return b.value - a.value});

               $.getJSON('languagesTopAndOthers.json', function (jsonC) {
                   legend.push(Object.keys(jsonC)[0]);
                   arrC = Object.keys(jsonC).map(function (k) {
                       return {name: k,value: jsonC[k]}
                   });
                   option = {
                       tooltip: {
                           trigger: 'item',
                           formatter: "{a} <br/>{b}: {c} ({d}%)"
                       },
                       legend: {
                           orient: 'vertical',
                           x: 'left',
                           data: legend
                       },
                       series: [{
                           name: 'C and others',
                           type: 'pie',
                           selectedMode: 'single',
                           radius: [0, '30%'],
                           center: ['50%', '56%'],
                           label: {
                               normal: {
```

```
                        position: 'inner'
                    }
                },
                labelLine: {
                    normal: {show: false}},
                data: arrC
            },
            {
                name: 'Other languages',
                type: 'pie',
                radius: ['40%', '55%'],
                roseType: true,
                clockwise: false,
                center: ['50%', '56%'],
                label: {small: {}},
                data: arr}]};
        var myChart = echarts.init(document.getElementById('chart3'));
        myChart.setOption(option);});});});
```

result: