CS5346 Mini-Project 1 GITViz

Student Name	**Dong Yizhou**	**Li Yuanda**
Matric Number	**A0078006J**	**A0078501J**

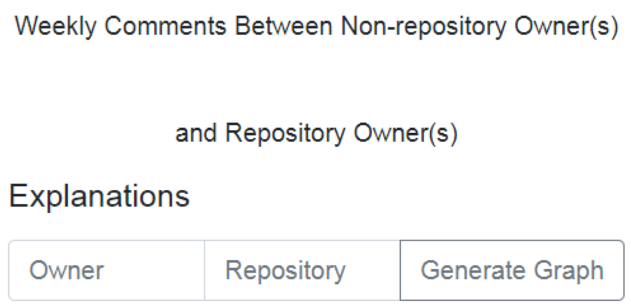
Introduction

Version control repositories are used in many IT related projects to keep track of project development progresses and retrieve critical project development information. To better access the data in the repositories, various visualization techniques are used to display versions, evaluation patterns and retrieve insights. Such techniques addressed user concerns about the project contribution styles, high-level building blocks and maintainability. In this mini-project, Yizhou and Yuanda took Torvalds repo data as a test case to develop a generic interactive visualization engine GITViz which support repositories data visualization. Yuanda did a deep research about Git API queries and develop tools to anticipate data extraction for generic repositories. Yizhou helped with data retrieving and processing. Both Yizhou and Yuanda contributed equally in the visualization creation.

Visualizations

Objective	**Visualization**
1	**Stacked Bar Chart**
2	**Heatmap**
3	**Donut Chart**

We developed a web app which automatically retrieve data requested for each objective. Users just need to fill in the name of the owner and the repository to auto generate the corresponding visualization.

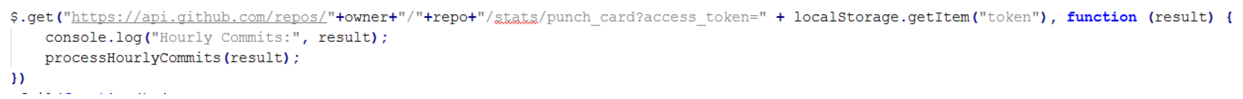


Visualization 1 – Stacked Bar Chart

Data Preparation:

We use data.js file to process data for all the 3 objectives and graph.js to generate visualization.

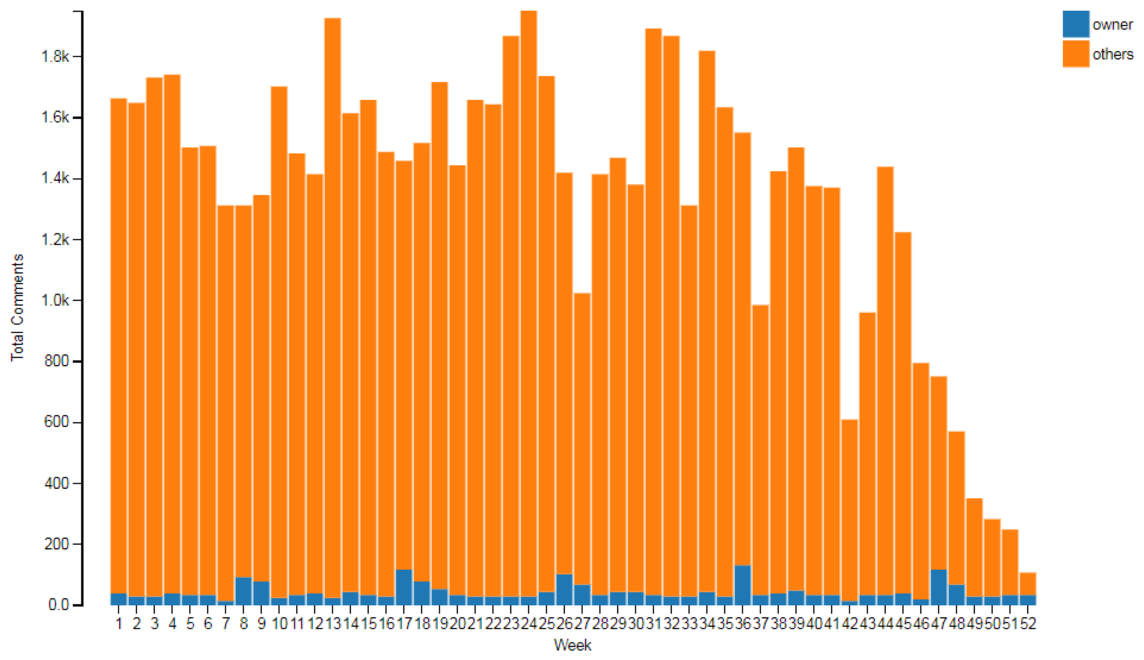
API below was called to retrieve the data for objective one where "owner" and "repo" are the customized value user filled in. Then `processParticipation` function was called to process the data and pass the variables including week counts, owner comment counts and non-owner comment counts to graph.js to generate the visualization.



```
$.get("https://api.github.com/repos/"+owner+"/"+repo+"/stats/punch_card?access_token=" + localStorage.getItem("token"), function (result) {
    console.log("Hourly Commits:", result);
    processHourlyCommits(result);
})
```

API: `GET /repos/:owner/:repo/stats/participation`

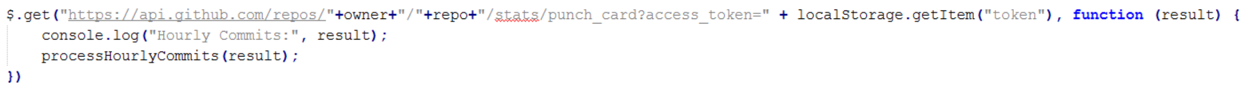
For objective one we choose Stacked Bar Chart as it is straightforward and users can view the different between weekly commit counts by non-repository owner(s) compared to repository owner(s) directly.



Visualization 1

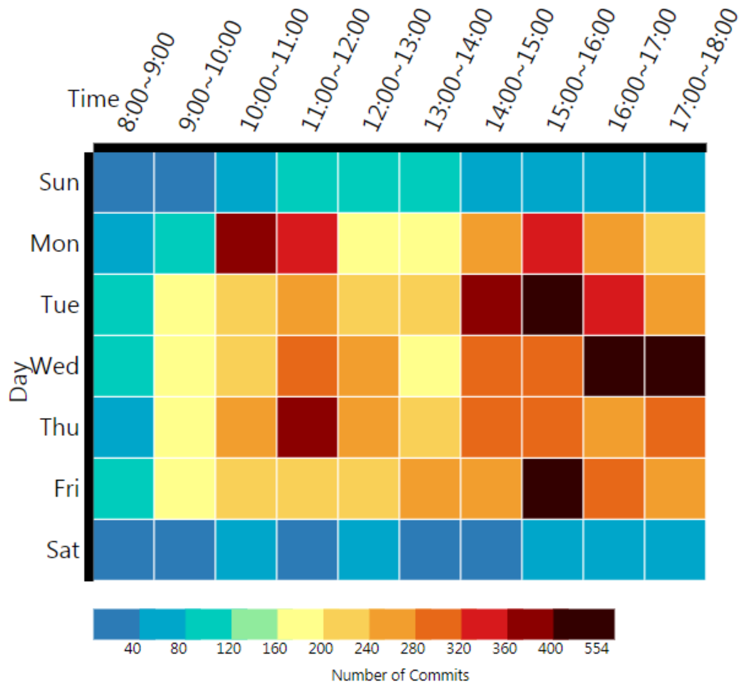
Visualization 2 – Heatmap

API below was called to retrieve the data for objective two. Then `processParticipation` function was called to process the data and pass the data to graph.js to generate the visualization. Then `processHourlyCommits` function is called to generate variables including days, hours and the total number of commit counts.



```
$.get("https://api.github.com/repos/"+owner+"/"+repo+"/stats/punch_card?access_token=" + localStorage.getItem("token"), function (result) {
    console.log("Hourly Commits:", result);
    processHourlyCommits(result);
})
```

API: `GET /repos/:owner/:repo/stats/punch_card`

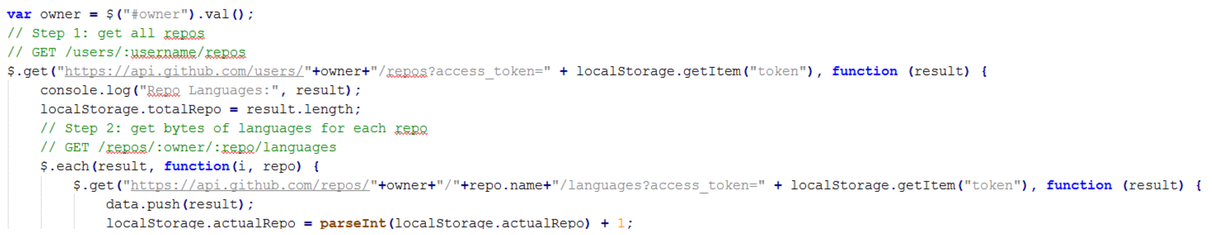


Visualization 2

For objective one we choose Heatmap as it reflects the comments distribution based on time and day.

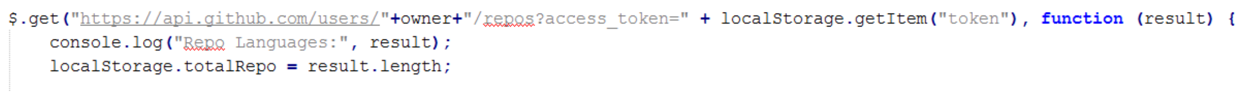
Visualization 3 – Donut Chart

2 APIs were called for objective three. The 1st API get all repositories belong to the specific owner the user filled in. The 2nd API calculate the total bytes of each language in all the repositories. Then several other functions are called to clean the data and pass the necessary info to graph.js for visualization.



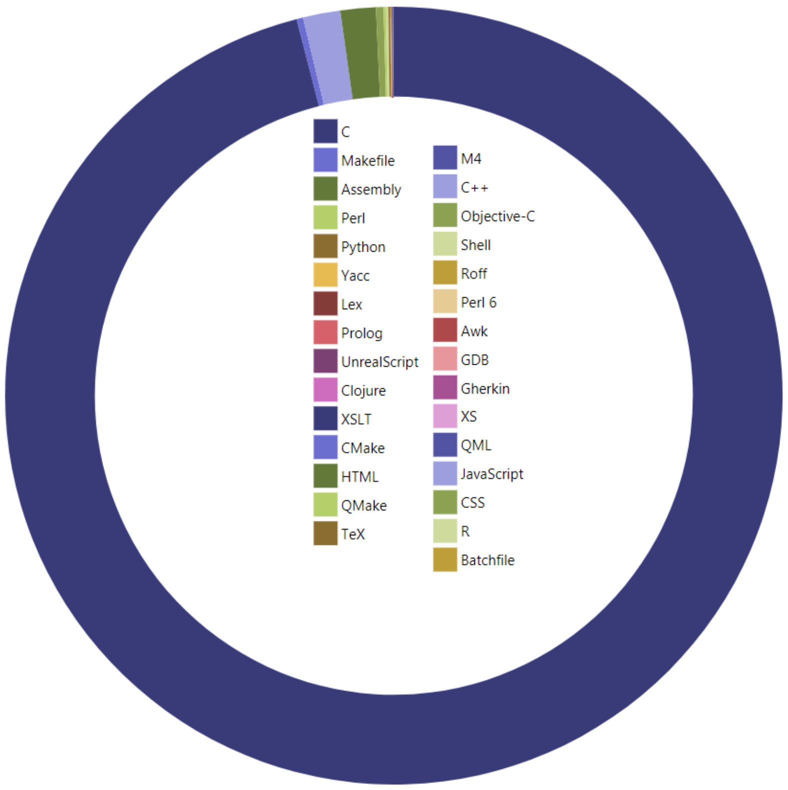
```
var owner = $("#owner").val();
// Step 1: get all repos
// GET /users/:username/repos
$.get("https://api.github.com/users/"+owner+"/repos?access_token=" + localStorage.getItem("token"), function (result) {
    console.log("Repo Languages:", result);
    localStorage.totalRepo = result.length;
    // Step 2: get bytes of languages for each repo
    // GET /repos/:owner/:repo/languages
    $.each(result, function(i, repo) {
        $.get("https://api.github.com/repos/"+owner+"/"+repo.name+"/languages?access_token=" + localStorage.getItem("token"), function (result) {
            data.push(result);
            localStorage.actualRepo = parseInt(localStorage.actualRepo) + 1;
```

API: `GET /users/:username/repos`



```
$.get("https://api.github.com/users/"+owner+"/repos?access_token=" + localStorage.getItem("token"), function (result) {
    console.log("Repo Languages:", result);
    localStorage.totalRepo = result.length;
```

API: `GET /repos/:owner/:repo/languages`



For objective three, Donut (Pie) Chart is the best chart to display the percentage of each category against total since the total bytes amount sum up to 100%.