Mini-Project-1 GIT(hub) Viz

Submission Date: Monday 12 March 10AM
Type: Pair work **Weightage: 15%**

I. Introduction

Version control repositories contain a wealth of implicit information that can be used to answer many questions about a project's development process. However, this information is not directly accessible in the repositories and must be extracted and visualized. Repositories covering long evolution periods, e.g., 5–10 years, provide interesting options for research on software evolution. By using dense pixel techniques, information on hundreds of versions of hundreds of files can be displayed on a single screen overview. Furthermore, interesting evolution patterns can be identified by directly looking at the visualization. Tuning various graphics parameters such as layout, texture, color, and shading yields different visualizations, thereby addressing different user questions : What is the contribution style in a given project? What are the high-level building blocks of a software system? How maintainable is a given project? What is the maintenance risk of a given developer leaving a given project? What has happened in this project while I was away? Which developers collaborate? The overall objective we have here is to develop a flexible and generic interactive visualization engine called GITViz that supports exploratory search in GIT repositories.

The purpose of the assignment is to:

1. **Extract a repo data e.g.** from **(https://github.com/torvalds/linux) , and visualize it for meaningful analysis.** The reference to APIs needed for data extraction is available at:
    - https://developer.github.com/v3/repos/statistics/
    - https://developer.github.com/v3/repos/

2. **Format data if needed.** The data you extract may not be in a concise format. It is often needed to be formatted/ organized e.g., using script (python) or using git --format.

3. **Use a visualization tool (e.g.** d3.js)

II. Important Information

1. Read this document carefully.

2. Use repo https://github.com/torvalds/linux as a test case for this assignment.

3. Demonstrate visualizations **to your tutor in the week of March 12** (a tutor availability schedule will be made available near to the time).

4. Submit a report (2-3 pages, single pdf), **as per Report template given at the end of this document, by Monday, March 12, 10 AM** in IVLE folder MP-GITViz-Report in IVLE Files(workbin). Exceeding the page guideline of 2-3 pages does not invite any

penalty.
Label the report document: GITViz_<Matric-number-1>_<Matric-number-2>
e.g. GITViz_A0045396X_A0046342Y.pdf

III. Task

Your task is to create **2 or more different type of visualizations to achieve the objectives given below**. Note that a single visualization does not have to cover all the objectives. **Each objective, however, should be covered by at least 1 visualization**.

There are in total **3 objectives:**

1. Visualize last 52-weeks' **weekly commit count by non-repository owner(s) compared to repository owner(s), starting from the most recent week.**

2. Visualize **total commits per working hour (8.00 am~6.00 pm) of each day (Monday to Sunday).** For example, *Monday 8 am has 50 commits; Monday 9 am has 25 commits… Sunday 5 pm has 10 commits, Sunday 6 pm has 30 commits.*

3. Visualize **the total bytes count of different programming languages used across ALL of specific author e.g. Torvalds' repositories.** Bonus point if your chart can anticipate future repositories that not yet exists.

Note that you can use any type(s) of visualization as long as you can achieve the above objectives. We value creativity.

You can **combine multiple API queries to get the data, and do extra processing (e.g., with python), and visualize the processed data with the tool of your choice**.

IV. What is git? Why do developer use it?

Git is a version control system used by a developer to manage their source code. Unlike typical storage service (i.e., Dropbox or Google Drive), Git keeps a detailed history the changes you made over time. Such feature is necessary when developers need to identify the person who made the changes or to resolve a conflict between two copies of implementations. It's also beneficial when you have multiple features in development so that your team can work on different "branches" concurrently and integrate on the same workspace. Last but not least, if it happens that you decide to abandon your current work, you can always "rollback" the changes smoothly without manual backtracking.

Installation

Windows user can download Git GUI here: https://git-scm.com/downloads.
Mac User can run the command in terminal: `brew install git`
Linux user should refer to their respective distribution guide on git installation. For example if you're on Ubuntu distro: `apt-get install git`

Git Basics

You can also refer to this interactive tutorial if you're interested in how Git works: https://try.github.io/levels/1/challenges/1

For the purpose of this assignment, you can focus on `git log` command. Here are some example commands you can try out:

List commits submitted since yesterday

- `git log --after="yesterday"`

List commits submitted between two dates

- `git log --after="2014-7-1" --before="2014-7-4"`

List commits submitted by specific author

- `git log --author="John"`

GIve Summary of authors and their respective commits

- `git shortlog`

Give detail changes (diff) between commits

- `git log -p`

Note that this is simply a introductory exercise. You do not need to extract information from command line. All required data can be extracted from Github API.

V. Data Processing

Github API provides live results in JSON format which is straightforward to use. However, you might find that some do not give you result you had in mind, therefore cannot be used directly in your visualization. For example, summing up total count across different arrays or filter unwanted information. Consequently, we need to do extra processing on the data.

Most programming languages provide methods that allow you to iterate through collections like List or Array easily such as standard iterative for-loop. There also exists another programming paradigm known as functional programming (FP). In this case, using FP, we can process collections more fluently. The following example uses Javascript.
Ref: https://code.tutsplus.com/tutorials/how-to-use-map-filter-reduce-in-javascript--cms-26209

map(): modify every element in collection in some way

```
var salaries = [1000, 1500, 2000, 2500];

// for each entry in salaries, we double the amount.
var doubleSalaries = salaries.map(salary => salary * 2);

console.log(doubleSalaries); // expected output: Array [2000, 3000, 4000, 5000]
```

filter(): filters out unwanted element given some conditions

```
var salaries = [1000, 1500, 2000, 2500];

// for each entry in salaries, if salary is more than 1700, we return the true (keep);
otherwise, we return false (discard).
var salariesOfMoreThan1700 = salaries.filter(salary => salary > 1700);

console.log(salariesOfMoreThan1700); // expected output: Array [2000, 2500]
```

reduce(): takes all of the elements in an array, and *reduces* them into a single value.

```
var salaries = [1000, 1500, 2000, 2500];

// for each entry in salaries, we sum them up.
var totalSalary = salaries.reduce((accumulation, salary) => {
       // use a curly bracket {} if you have multiple line in your processing body
       return accumulation + salary;
}, 0); // don't forget the identity/initial value of accumulator!

console.log(totalSalary); // expected output: 7000
```

Putting it together: Method Chaining, define a series of steps to transform the data

```
var salaries = [1000, 1500, 2000, 2500];

var processedSalaries = salaries.map(salary => salary *2) // [2000, 3000, 4000, 5000]
                                .filter(salary => salary > 3500) // [4000, 5000]
                                .reduce((accumulation, salary) => {
                                    return accumulation + salary;
                                }, 0); // 9000

console.log(processedSalaries) // expected output: 9000
```

VI. Report Template

Mini-Project: GIT(Hub) Viz

Student Name		
Matriculation Number		

1. Introduction

(up to 1 paragraph including objective of assignment in your own words; individual contribution of each member in doing this assignment)

2. Visualizations - Purpose & Method

(i) State which visualization(s) did you select for each of the objectives given in Section III. ITo facilitate grading, you can use a table showing which objectives are covered by which visualization. *An example is given below:*

Objective	*Visualization*
1	*Heatmap*
2	*PieChart, Heatmap*
3	*Waterfall*

(ii) For each of visualizations provide :

- A stepwise method you followed in creating the visualization. Be precise and succinct. Include GitHub API/ scripts you used. Write with a perspective such that your peers could easily use your method to create a similar visualization.
- An image of the visualization

3. (optional) Any other comments or information you may have
