Mining MatPlotLib

Dylan Klintworth, Stephanie Rodriguez, Alvin Alic, Alberto Cattaneo, Matt Wozniak

Code Overview

Main Libraries:

- PyGitHub, Github3
- Pandas or CSV

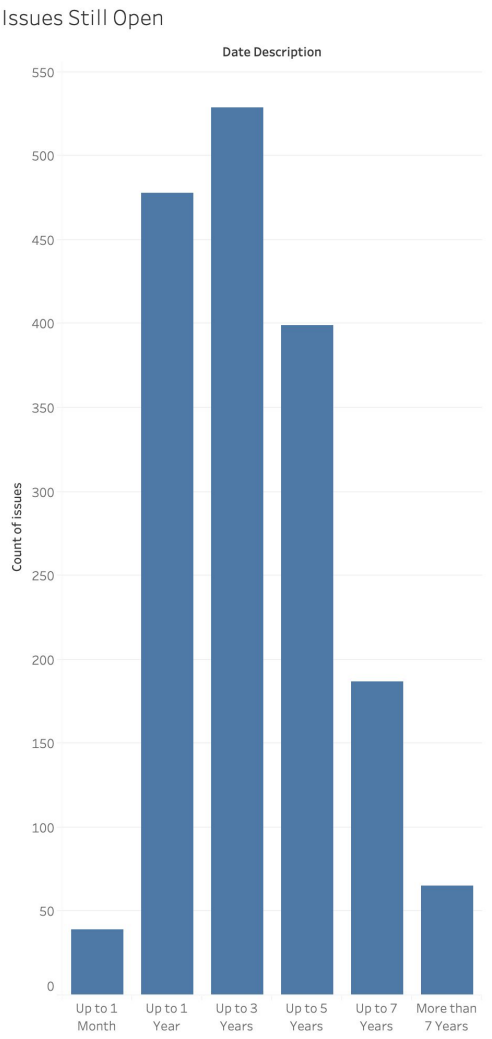
Blockers:

- API Limits
- 5,000 API calls per hour per user

```
import datetime, copy, csv
from github import Github # pip install PyGithub
import pandas as pd # pip install pandas

'''
GETTING STARTED WITH PYGITHUB

from github import Github
# First create a Github instance using an access token:
g = Github("access_token")
# Then play with your Github objects:
repo = g.get_repo("matplotlib/matplotlib"):

WHEN CREATING PERSONAL ACCESS TOKEN
Go to github.com/settings/tokens
Click on Generate new token
Select public_repo
Copy token into code below
'''

def main():
    '''
    Runs the main functions
    '''
    start  = datetime.datetime.now()
    g      = Github("token")
    repo   = get_repo(g)
    issues = get_all_issues(repo)
    get_issues_over_time(issues)
    end    = datetime.datetime.now()
    print(f"Completed in: {end-start}")

def get_repo(tokenized_github):
    """
    Returns the Repository object representation of the matplotlib repository

        Parameters:
            github (github.Github): The main GitHub object representation
    """
    repo = tokenized_github.get_repo("matplotlib/matplotlib")
    return repo
```

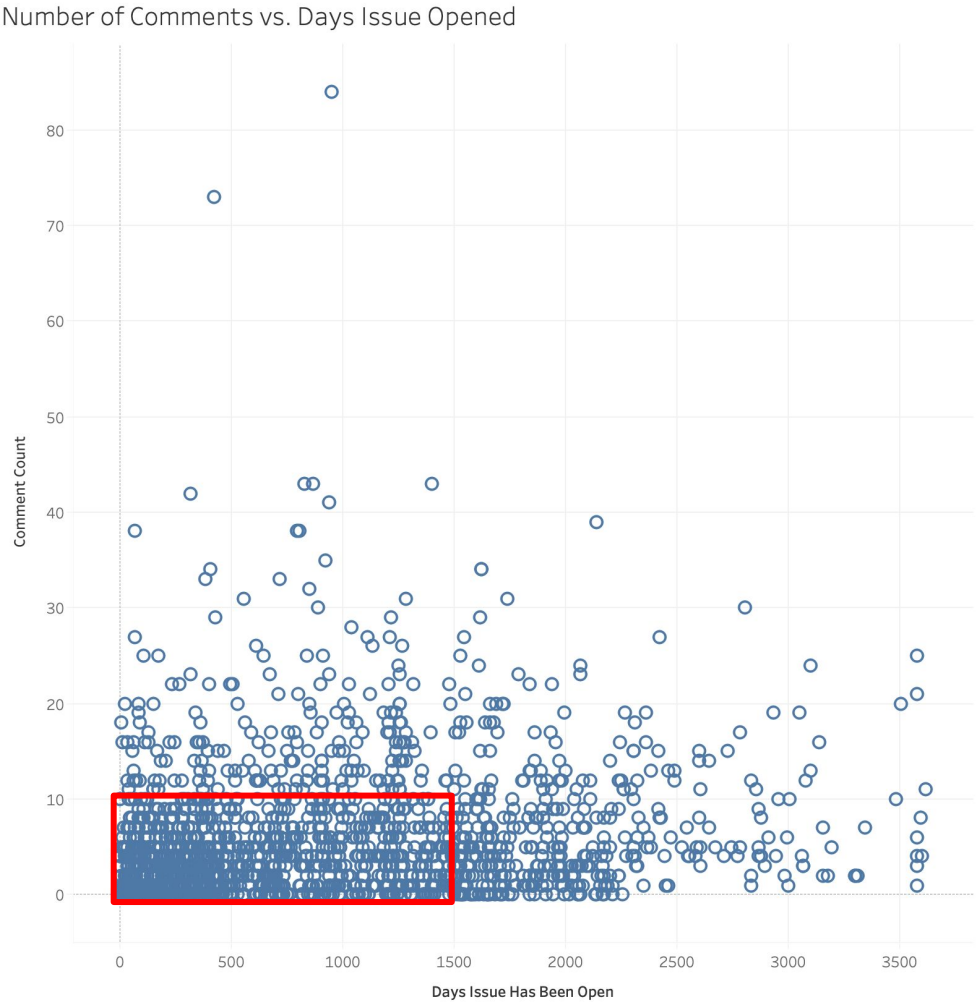
Issues

Most issues that are opened have been left open for 1 - 5 years.



Issues

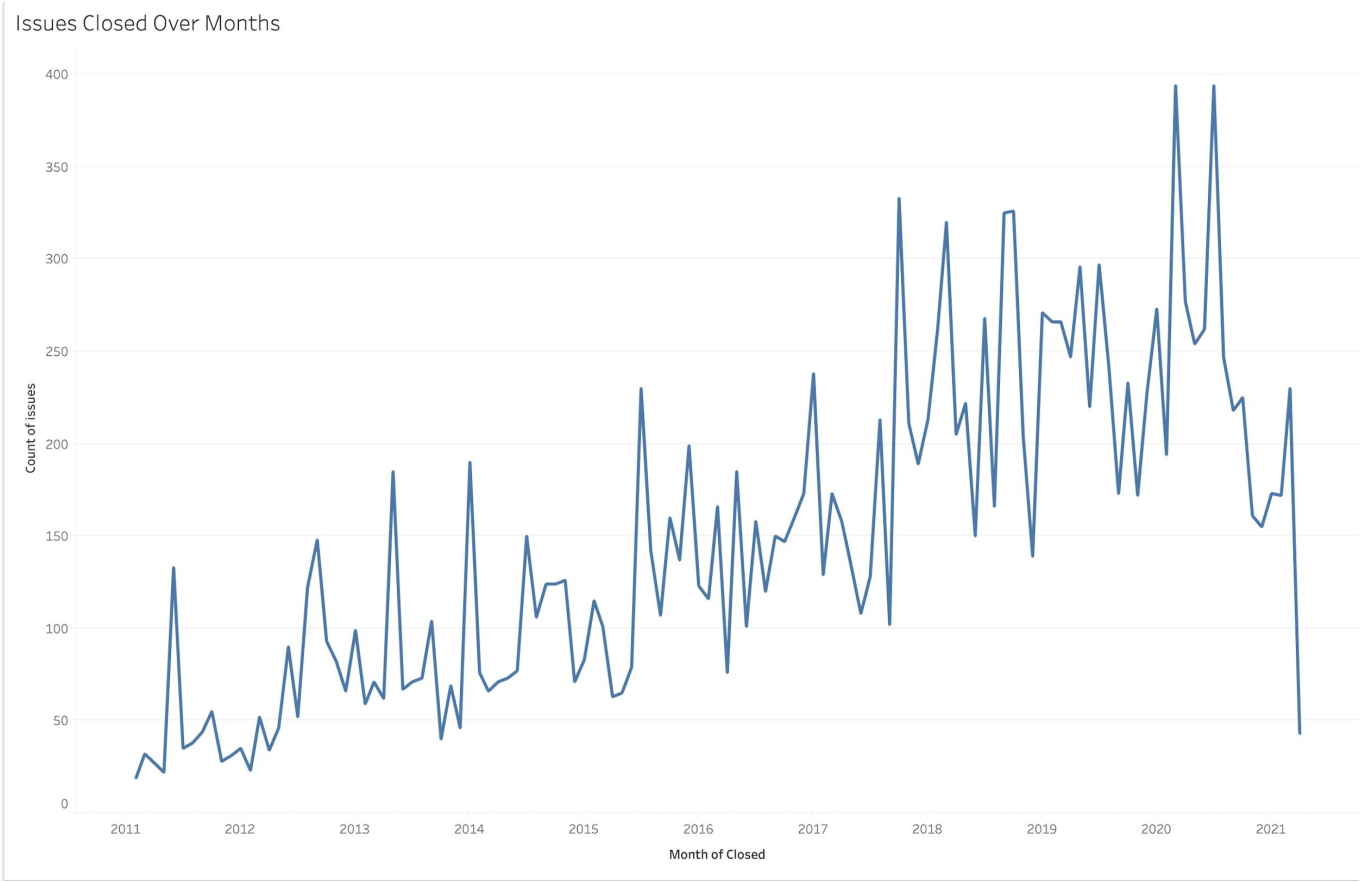
- No correlation between the number of comments vs. days issue opened.
- Most dense area appears to be in Days Open < 1,500 and Comments < 10.
- No analysis conducted on assignees as the majority of issues contained 0 assignees.



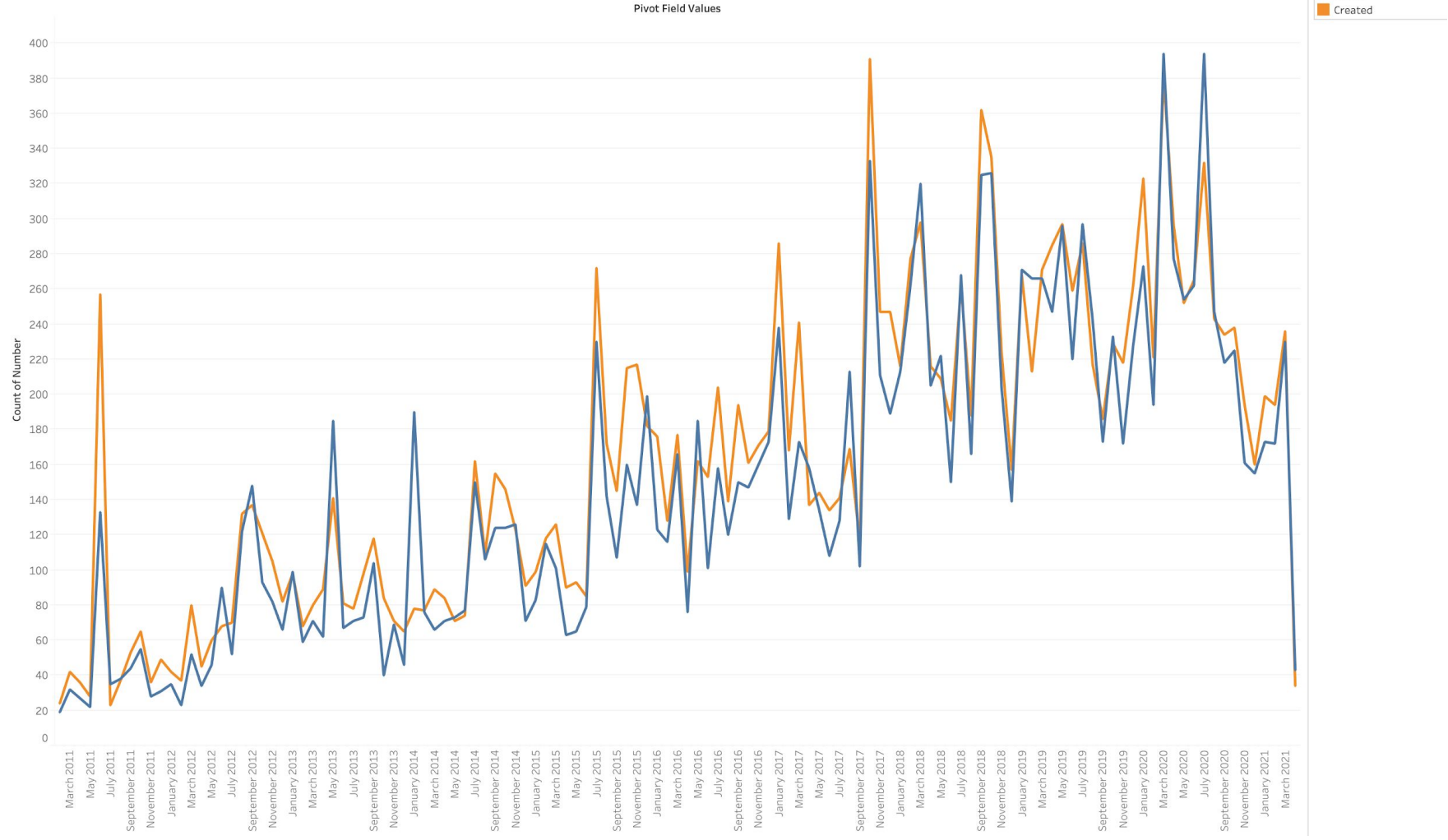
Issues

Most active months for closing issues (at 394):

- March 2020
- July 2020

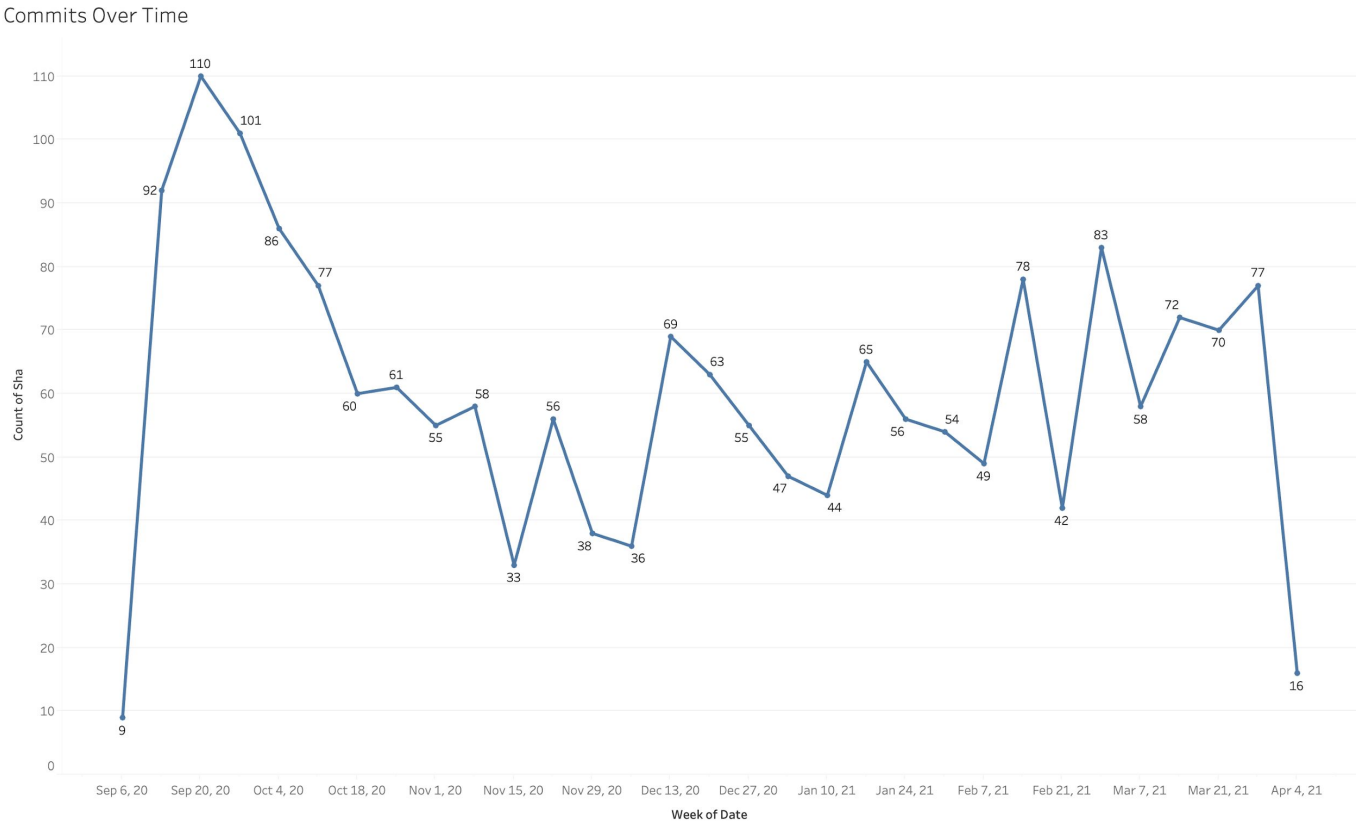


Closed Vs. Created Issues



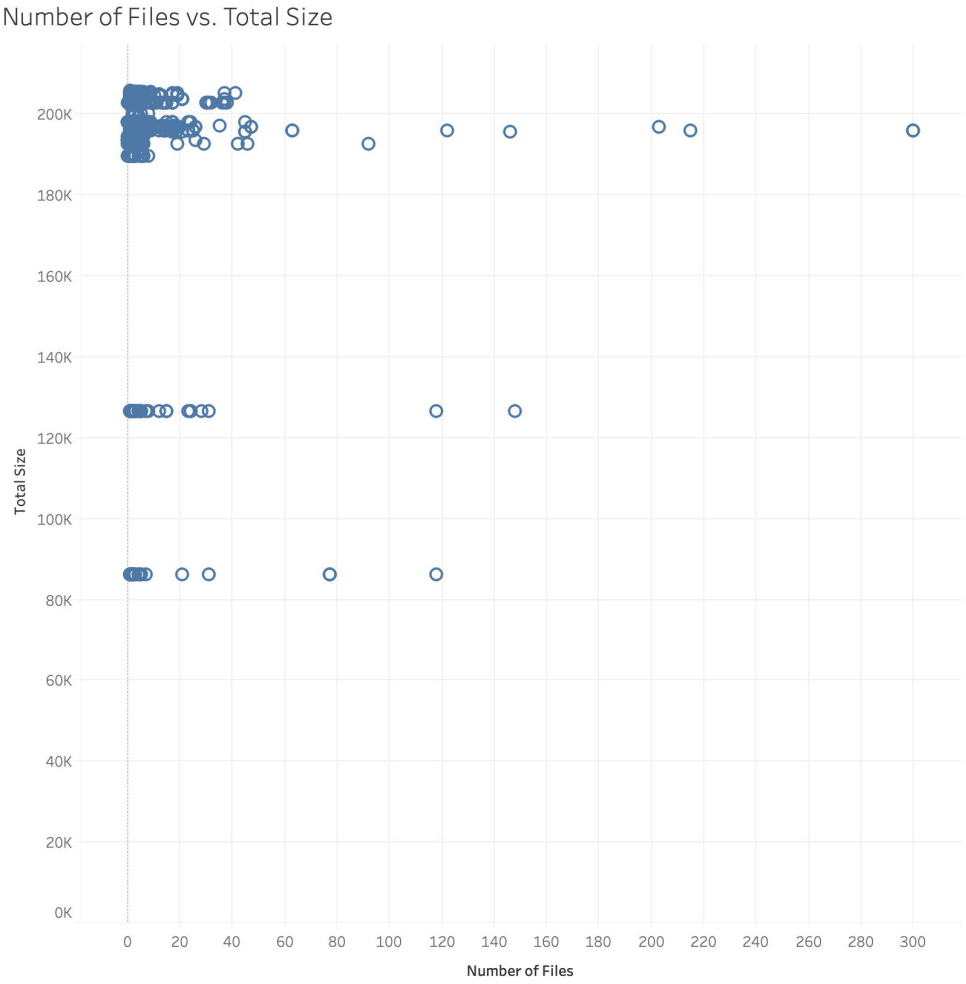
Most Recent 2,000 Commits

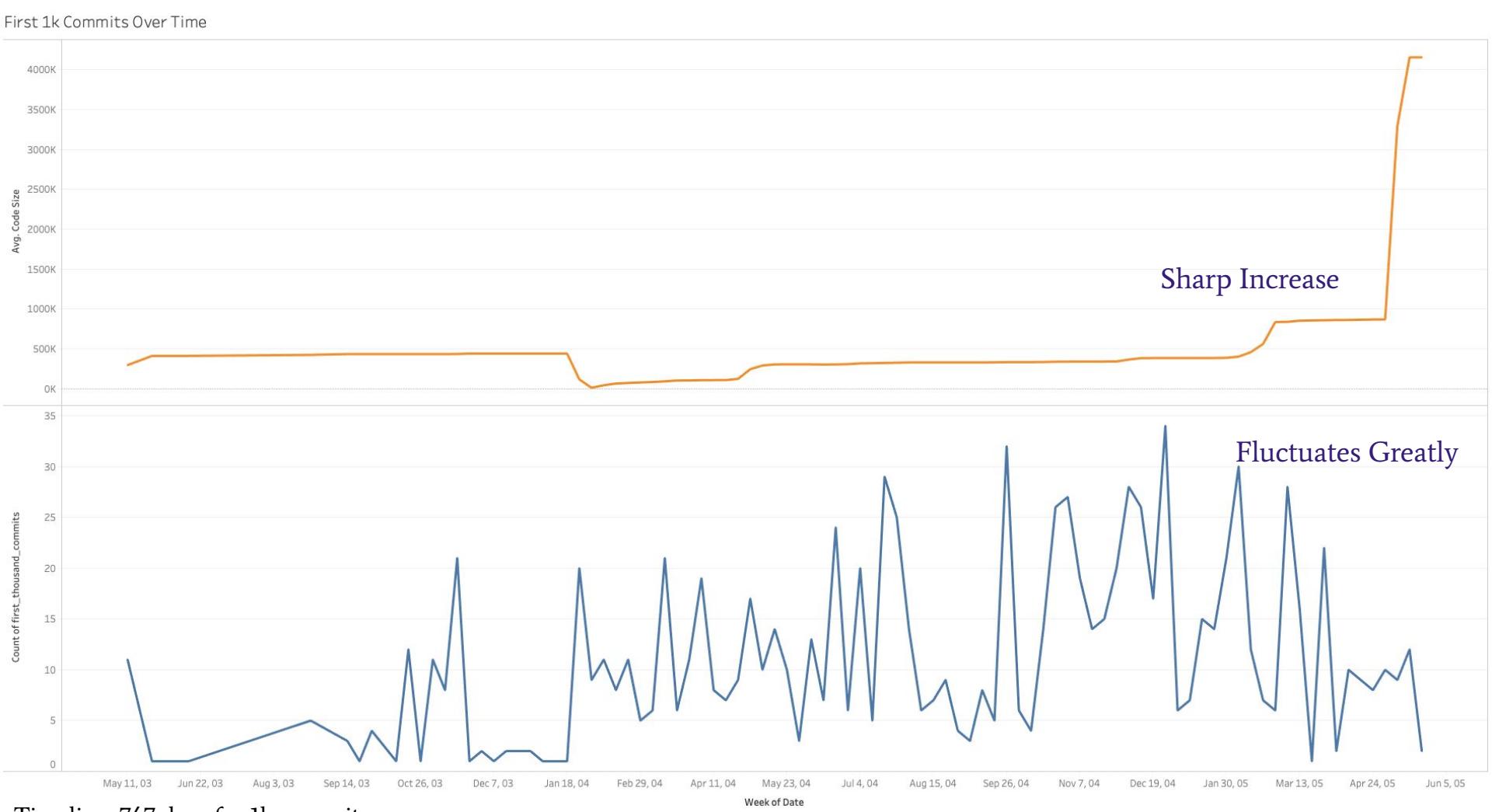
- Week of most commits: Sept 20, 2021 with 110 commits
- v3.3.2 was released September 15, 2020
- Assumption: Most peaks will correlate with releases



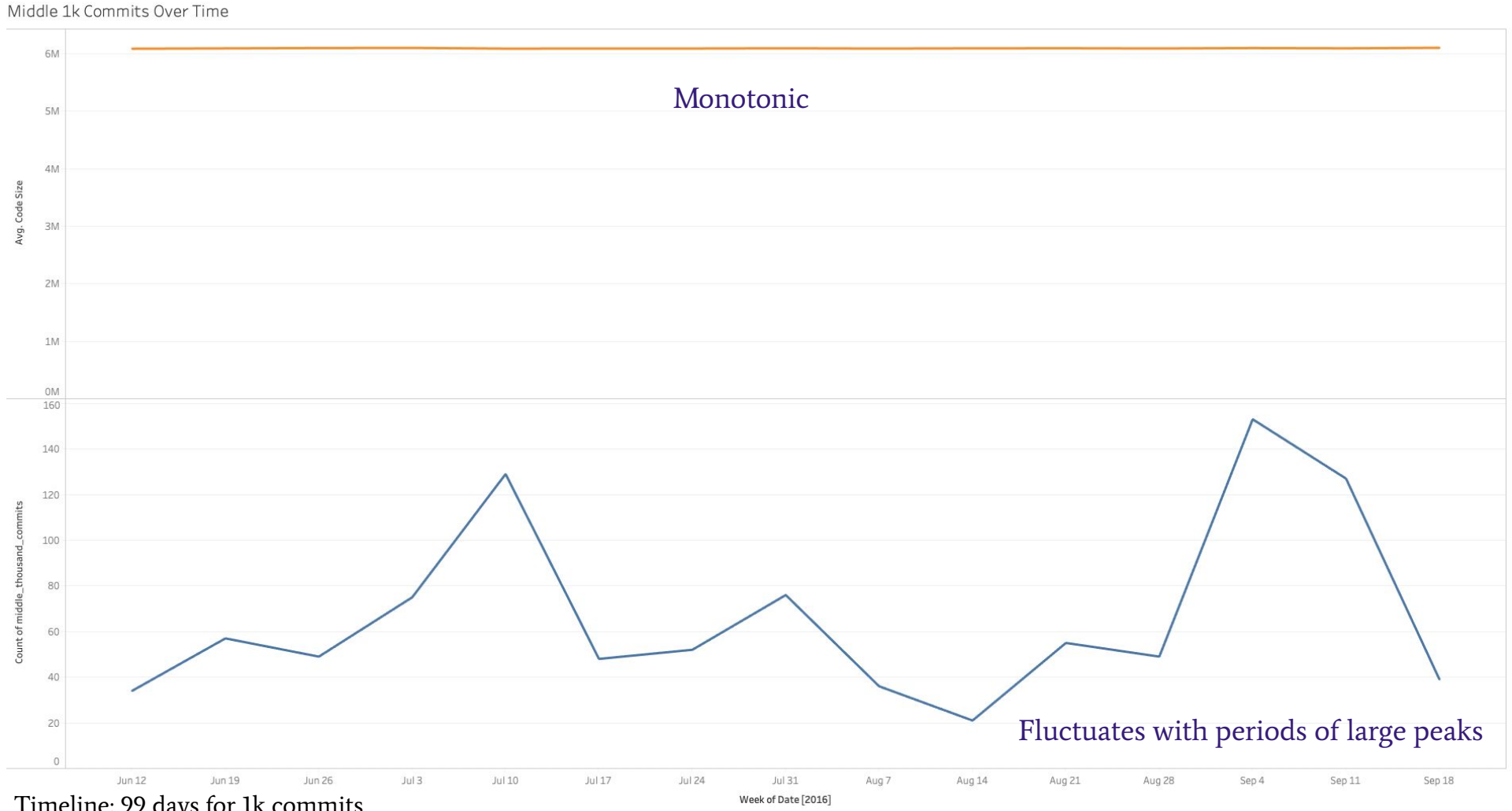
Most Recent 2,000 Commits

- The file size seems to be consistent, even with the addition of new files
- Might suggest new files are a result of refactoring (e.g., splitting code into different files, etc.), especially as the project is more mature (last 2k commits)
- Or there is a strict project structure and most contributors only add to existing files

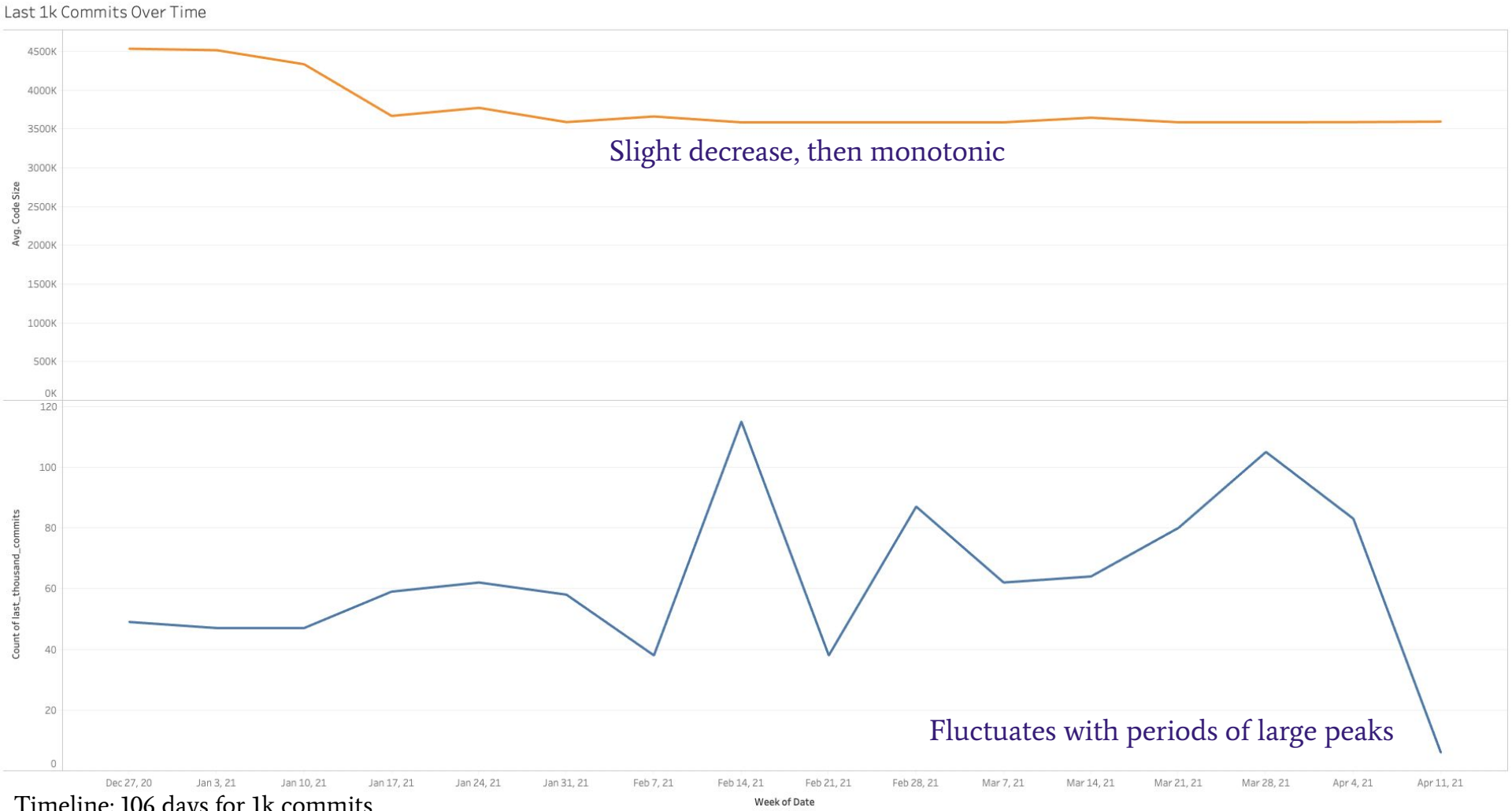




Timeline: 747 days for 1k commits



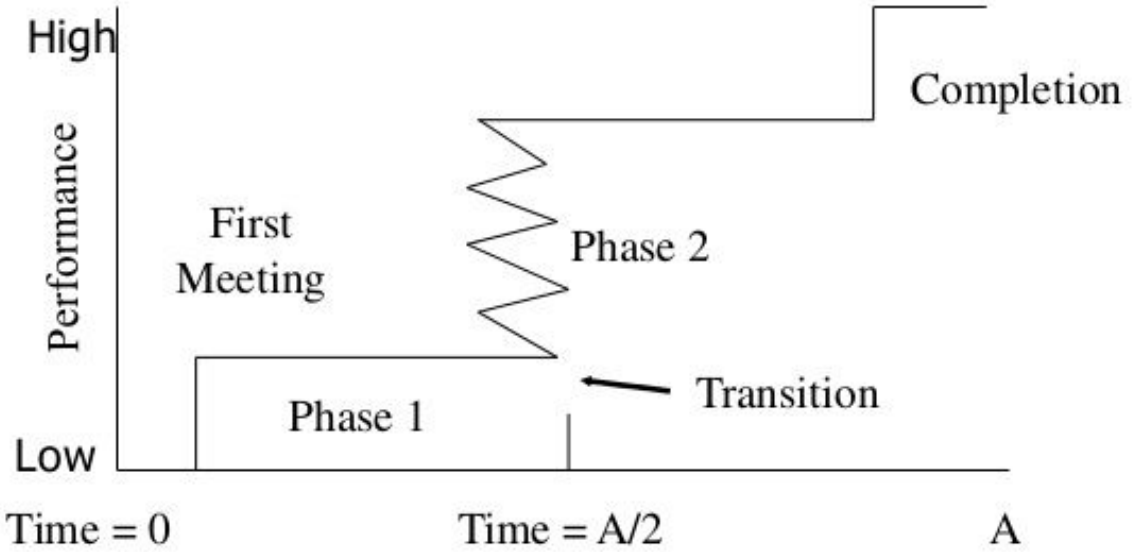
Timeline: 99 days for 1k commits



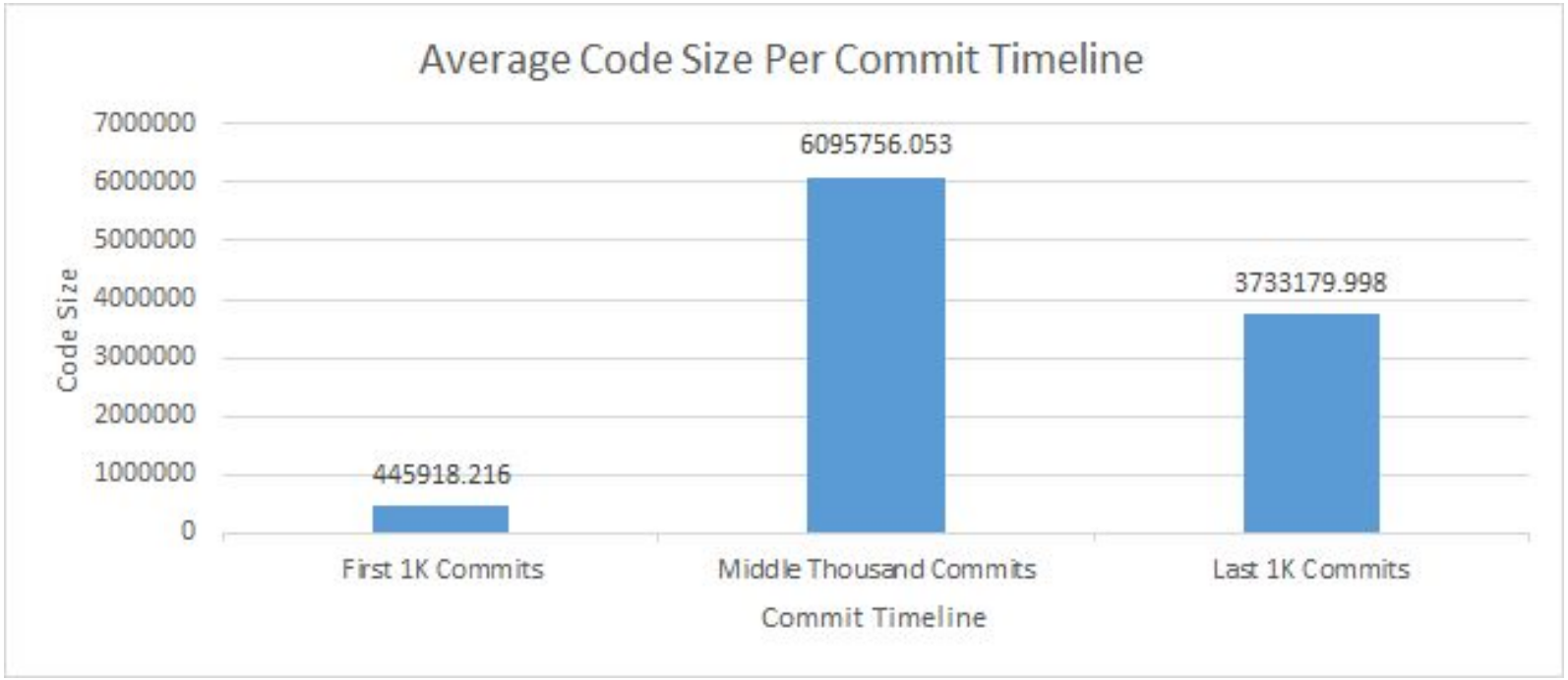
Timeline: 106 days for 1k commits

MatPlotLib - Group Performance

Periodic sharp increases corresponds with the well-known **Punctuated Equilibrium Model** for group projects. Most effort is at time A (end) of a project, in this case: version releases.

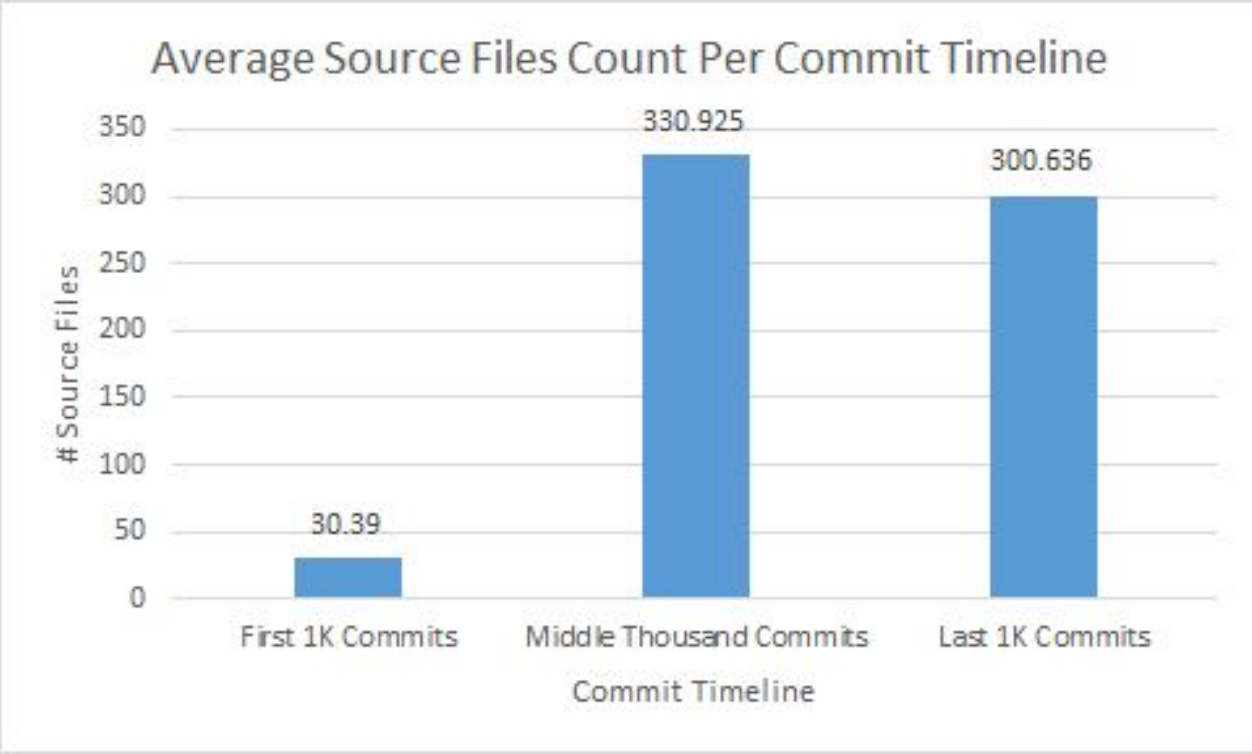


Commits - Code Size



Source Files Tracked: .c, .cpp, .css, .h, .html, .js, .m

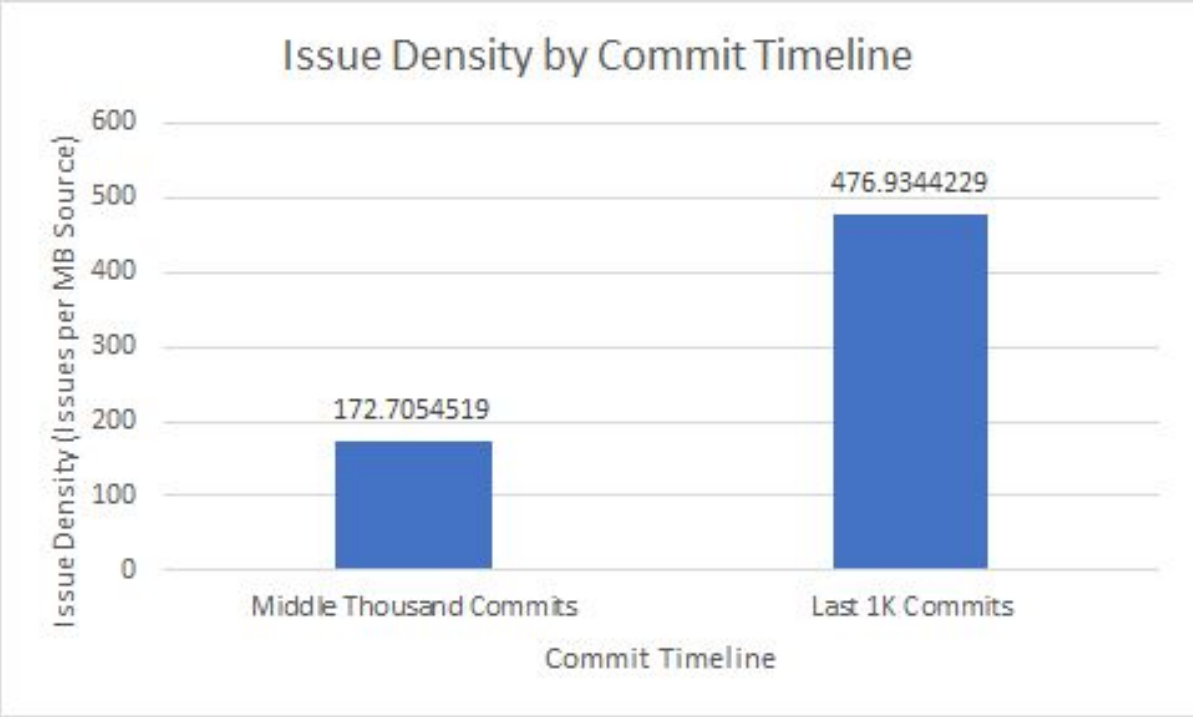
Commits - Source File Count



Commits - Code Size Analysis

- The domain of the code size analysis was the set of commits used within a specific period.
    - E.g: First thousand commits timeline was the first thousand commits of the matplotlib repository.
    - Each commit had a code size associated for each of all source files.
- The codomain of the code size analysis could have been all real numbers up to 100GB as it is documented by GitHub that repositories cannot exceed 100GB in size.
- The function used is a total function because it accounts for all possible inputs. If the commit timeline has 1...N commits, it will compute the average of the code size for all commits within the set.
- The function is continuous as every input of a commit set produces an output.
- The output fluctuates between each given commit timeline as the average code size in each set grew or shrunk from timeline to timeline.
- The code size shrunk 38% from the middle thousand commits to the last thousand commits even though the quantity of source files only decreased by 9%.

Issue Density



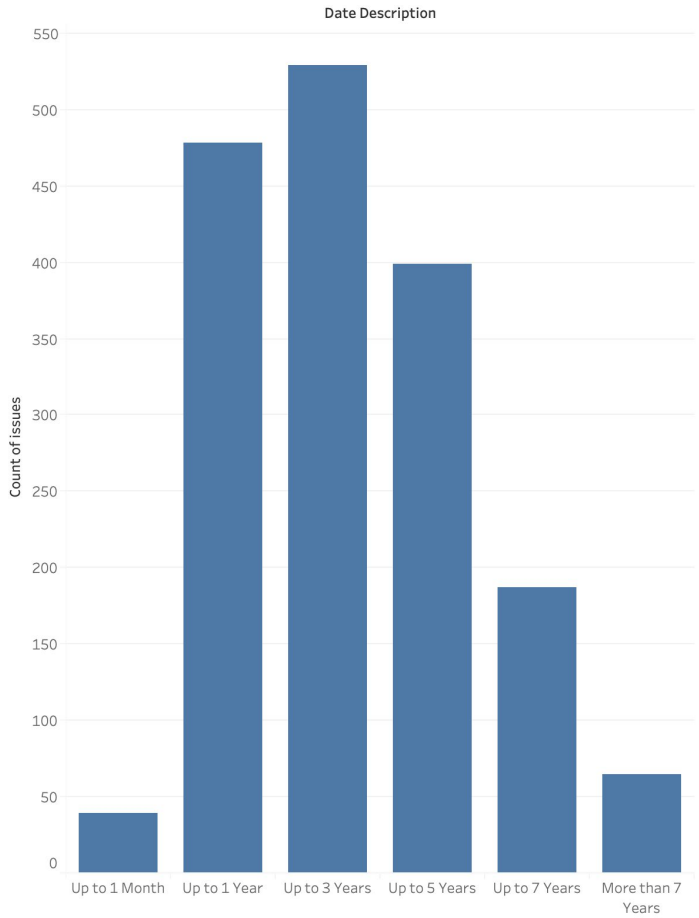
Issue Density Analysis

- The domain of the issue density analysis was the average code size of a commit set and the number of unclosed issues up to the end of that commit set's timeline.
    - E.g: 6095756 avg code size, 1004 open issues.
- The codomain of the issue density analysis could be 0 to all real positive numbers.
- The function used is a partial function because it doesn't account for all possible inputs. If the average code size input was 0, then the function would be undefined.
- The function is continuous unless the input is zero for the average code size.
- The output fluctuates as more issues are added and as code size changes throughout the commit timeline.
- As the code size depleted by the last thousand commits, the issue density increased by almost three times that of the middle thousand commits.
    - As commit frequency decreases, more issues have been found.

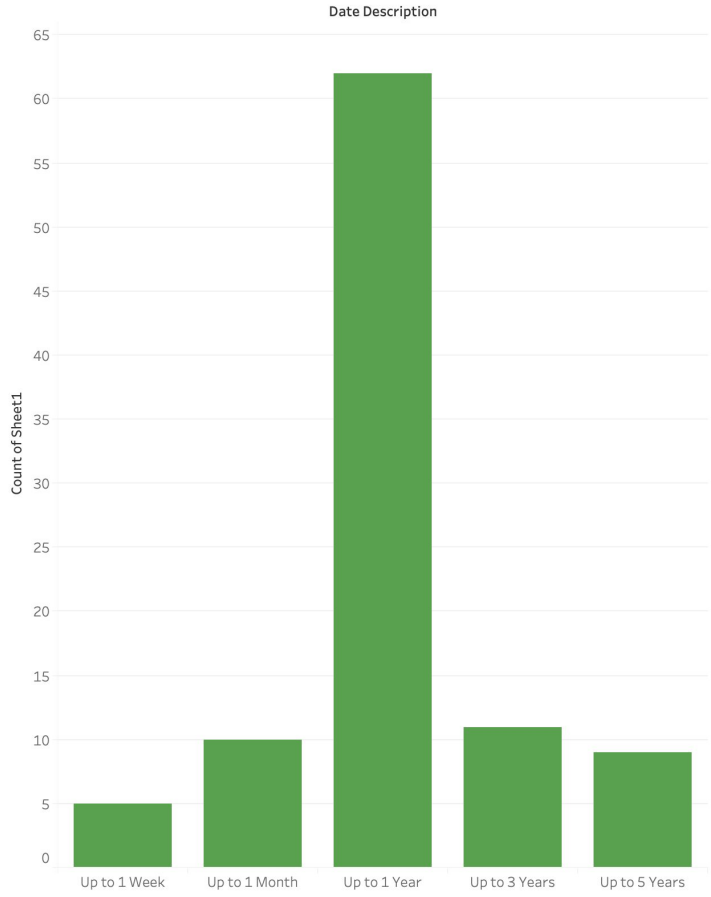
PyGitHub Comparison

MatPlotLib vs. PyGitHub

Issues Still Open



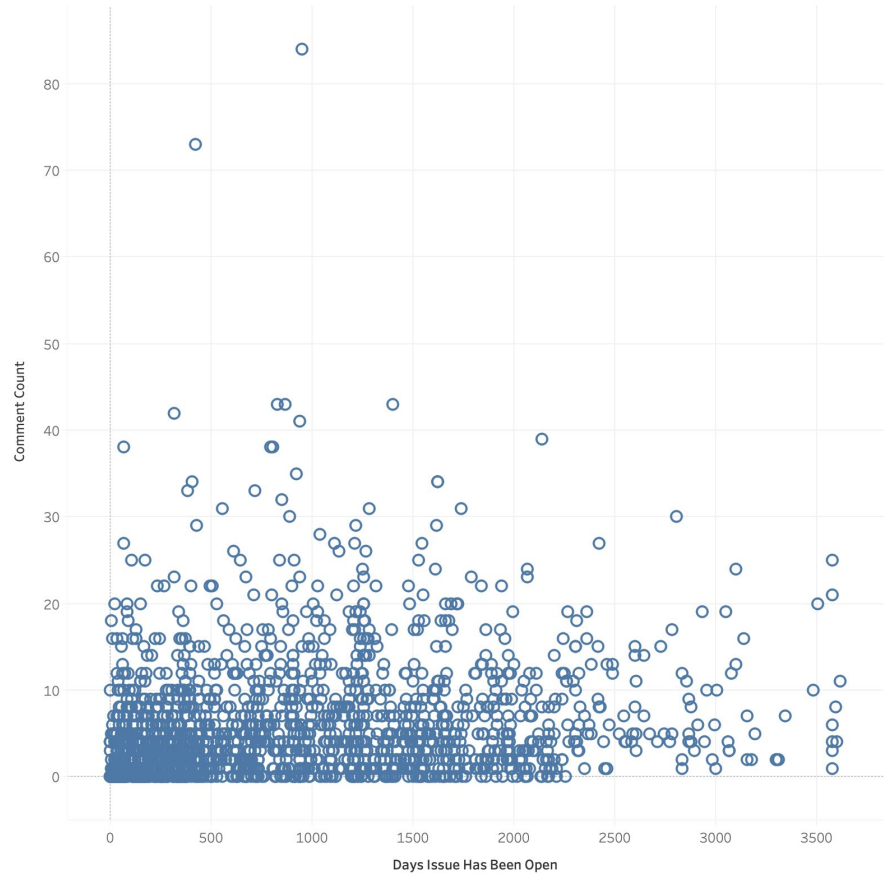
Issues Still Open



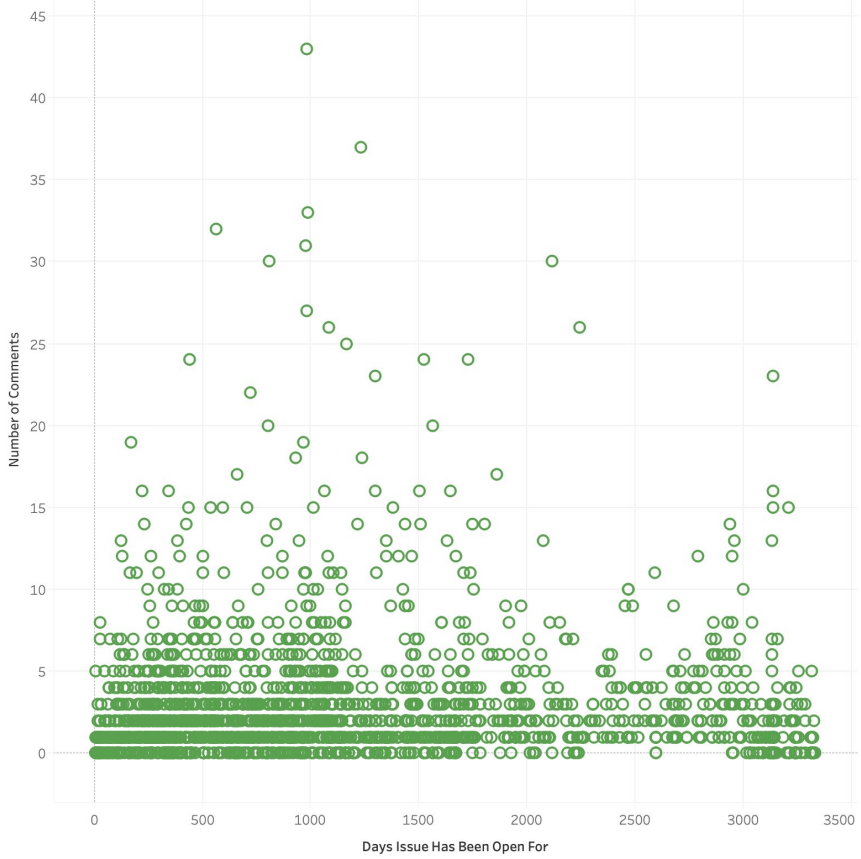
MatPlotLib vs. PyGitHub

*PyGitHub also did not really utilize the Assignee feature

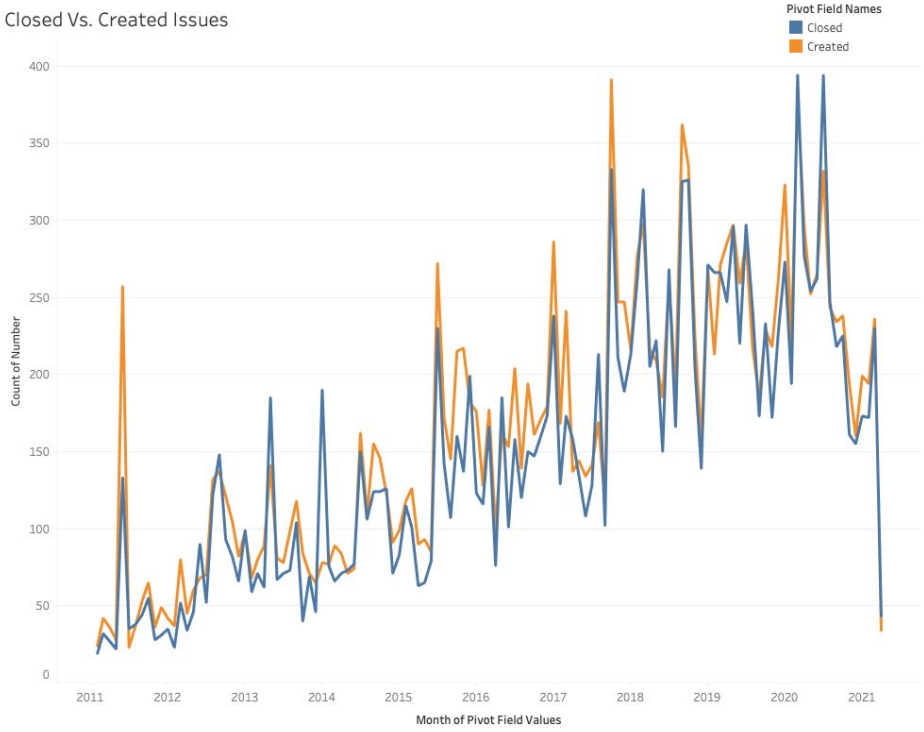
Number of Comments vs. Days Issue Opened



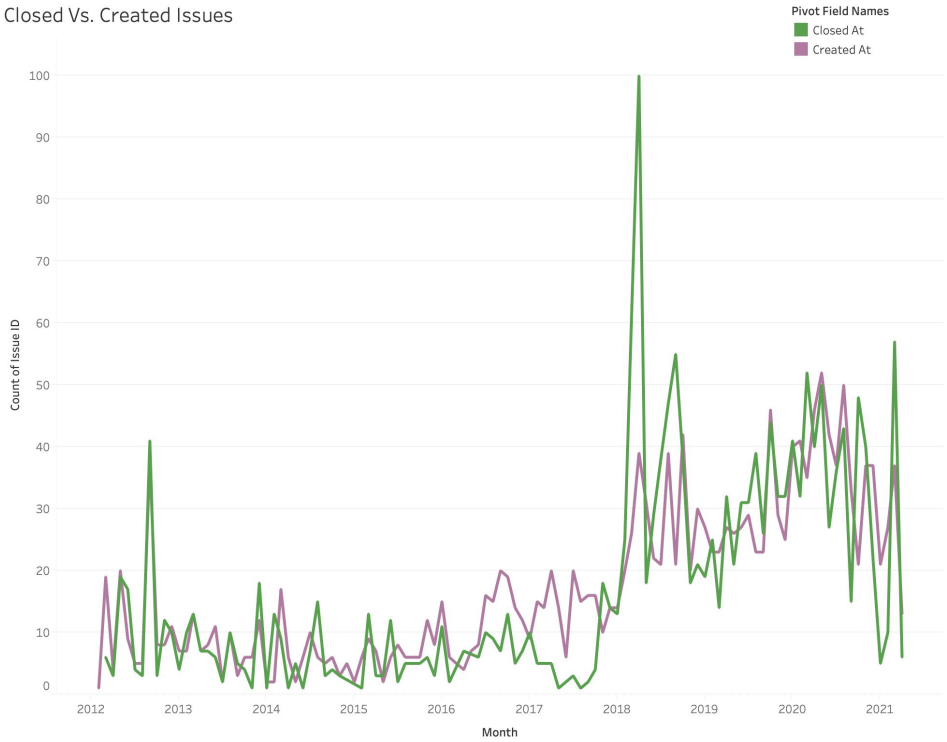
Number of Comments Vs. Days Issue Still Open



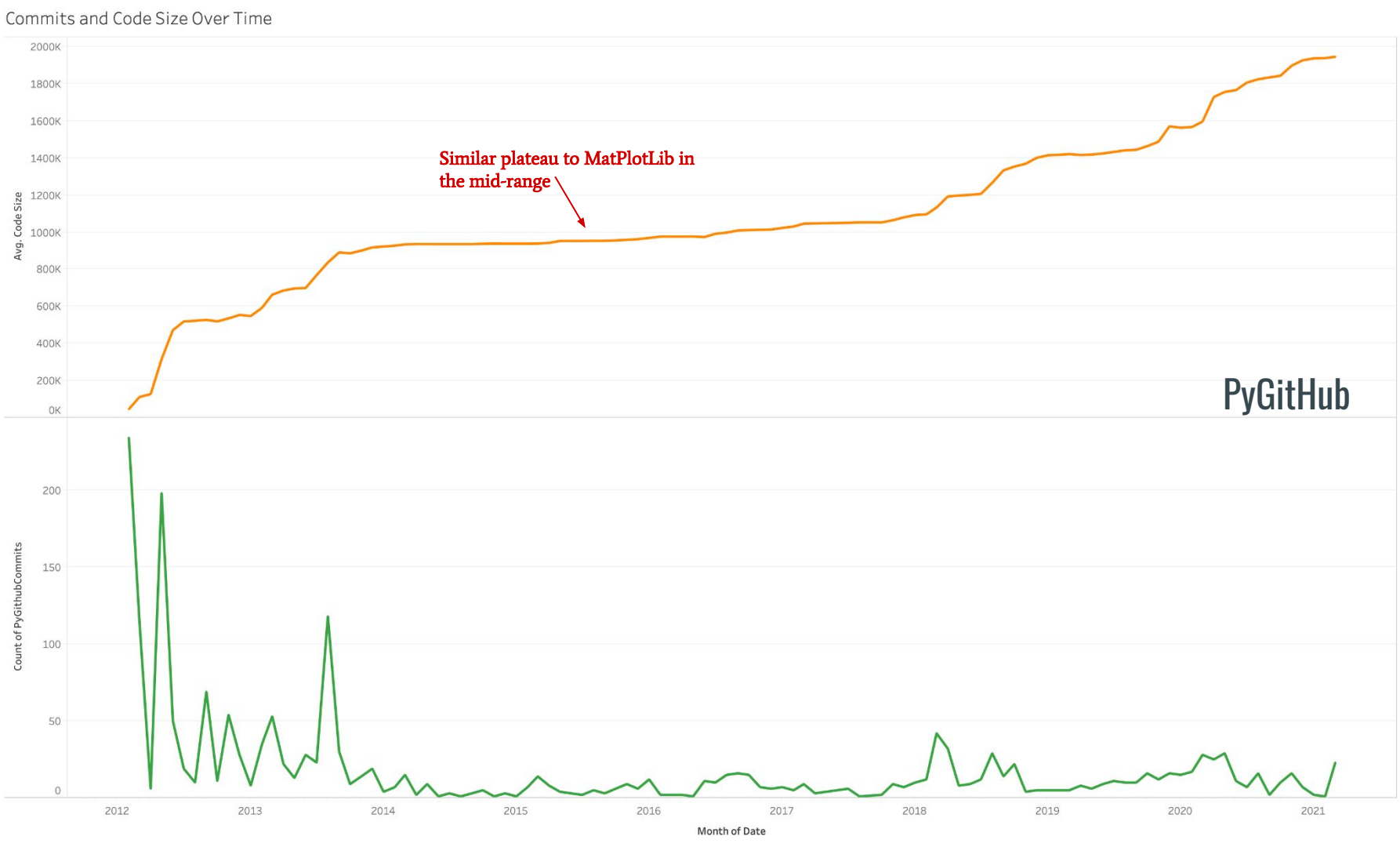
MatPlotLib vs. PyGitHub



More sharp increases of created issues.



More sharp increases of closed issues. Corresponds to Issues Still Open bar chart differences.


Commits and Code Size Over Time
Avg. Code Size
2000K
1800K
1600K
1400K
1200K
1000K
800K
600K
400K
200K
0K
Similar plateau to MatPlotLib in the mid-range
PyGitHub
Count of PyGithubCommits
200
150
100
50
0
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
Month of Date