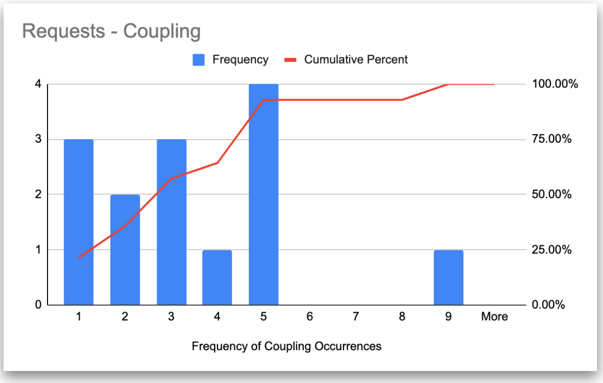
Requests (https://github.com/psf/requests)

We desire low coupling, high cohesion. Areas with zero cohesion should be areas of concern and looking into it.

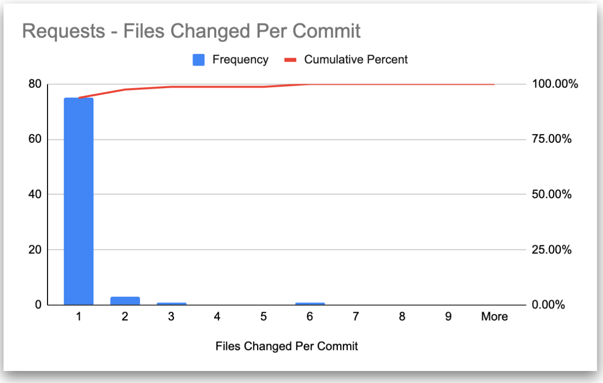
⚠️ Coupling

Desire low coupling.



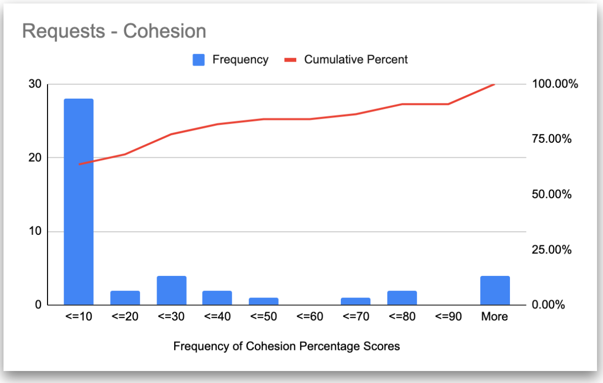
✅ Files Per Commit

Desire low number of files per commit.



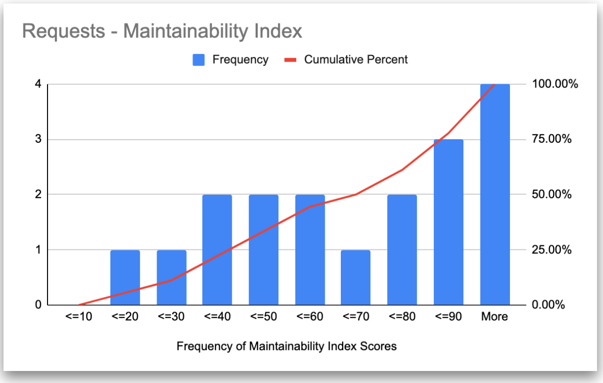
‼️ Cohesion

Desire high cohesion.



✅⚠️ Maintainability Index

Desire maintainability index above 20.

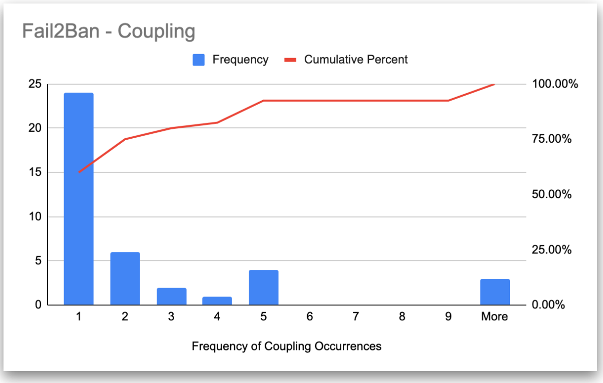


Fail2Ban (https://github.com/fail2ban/fail2ban)

We desire low coupling, high cohesion. Areas with zero cohesion should be areas of concern and looking into it.

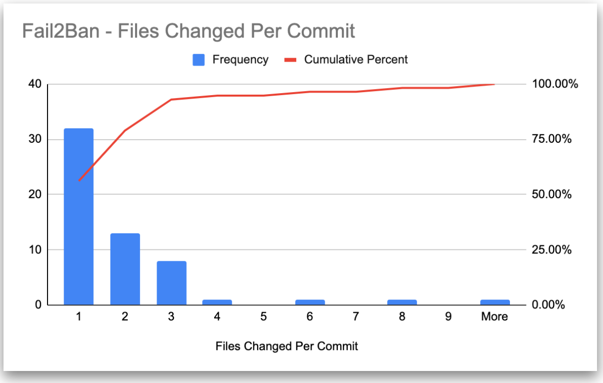
✅ Coupling

Desire low coupling.



✅ Files Per Commit

Desire low number of files per commit.



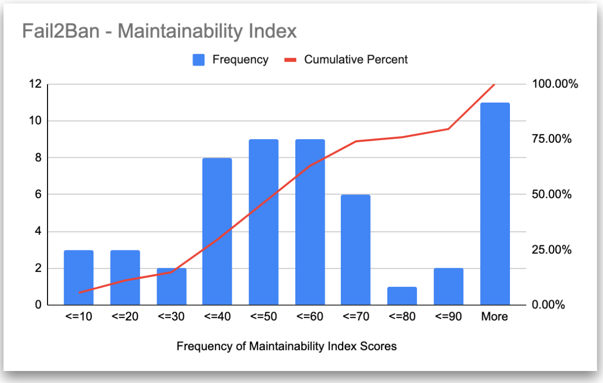
❓ Cohesion

Desire high cohesion.

UNABLE TO RUN THE COHESION SCRIPT ON THIS REPOSITORY.

✅⚠️ Maintainability Index

Desire maintainability index above 20.



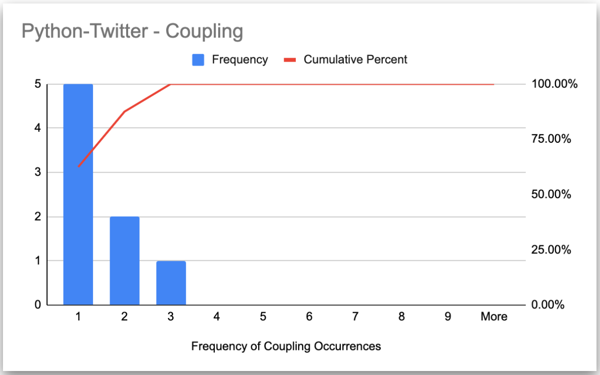
Python-Twitter (https://github.com/bear/python-twitter)

We desire low coupling, high cohesion. Areas with zero cohesion should be areas of concern and looking into it.

The files in the "More" range for **files per commit** are incredibly high, often 50, 90, or 100 files!

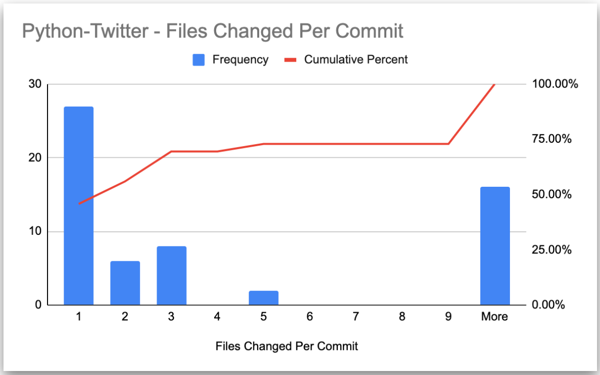
✅ Coupling

Desire low coupling.



‼️ Files Per Commit

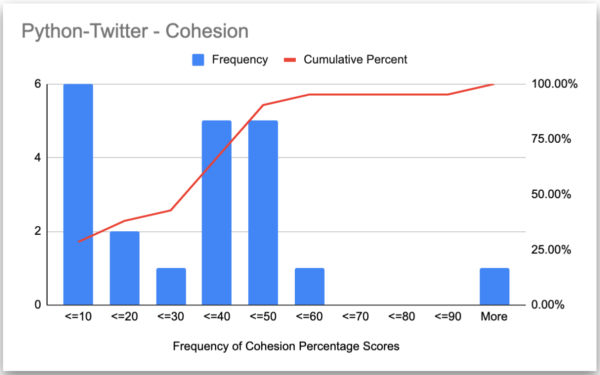
Desire low number of files per commit.



THE FILES IN THE "MORE" RANGE ARE INCREDIBLY HIGH, OFTEN 50, 90, OR 100 FILES!

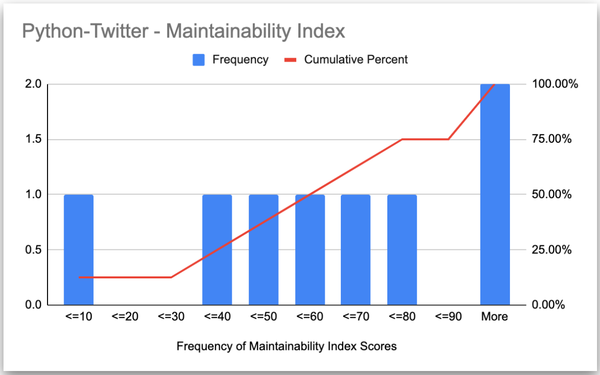
‼️ Cohesion

Desire high cohesion.



✅⚠️ Maintainability Index

Desire maintainability index above 20.



Summary of Findings

The three repositories I reviewed were **Requests** (https://github.com/psf/requests), **Fail2Ban** (https://github.com/fail2ban/fail2ban), and **Python-Twitter** (https://github.com/bear/python-twitter). I found these to be of similarly ranked size (and not too far off in the number of commits) in the project list, providing metrics to comparable projects.

When reviewing the metrics for **coupling** on each project, all tended towards the lower end, which is desirable. The **Requests** repository had the highest numbers on its low-end dependency frequencies, with dependencies imported into files five times having the most frequency. Considering how small these projects are, I wonder if we can improve this number for this repository. **Fail2Ban** does much better, with single dependency imports much more common and **Python-Twitter**, which never exceeds a score of 3.

Related to the coupling scores, we should also consider the **cohesion** for a project. While **Fail2Ban** had something in its code that made the existing script unable to run as-is, we could still obtain cohesion scores for the files in **Requests** and **Python-Twitter**. We desire to have low coupling (we did okay there!) And high cohesion. However, both projects leaned more in the direction of low cohesion. **Requests** had nearly 70% of its code with a score of 20% cohesion or less! **Python-Twitter** did slightly better with almost 70% of its score at 40% cohesion or less. Both code bases could use some review in cohesion to see if there are places to improve the cohesion scores.

Files Per Commit is a metric that may be more related to code culture for that open source repository than the codebase itself but is still worth reviewing. Generally speaking, we want to aim for small changes and a few files touched in every commit. Single-file commits can indicate low coupling. Smaller commits make it easier to roll back if needed and are much easier to review. **Requests** did very well in this area, with 93.75% of commits editing only a single file! **Fail2Ban** wasn't far behind, with 93% of its commits impacting only three files or less; only 56% of commits affected only a single file. **Python-Twitter** was also in a pretty good position, with 69% of commits impacting

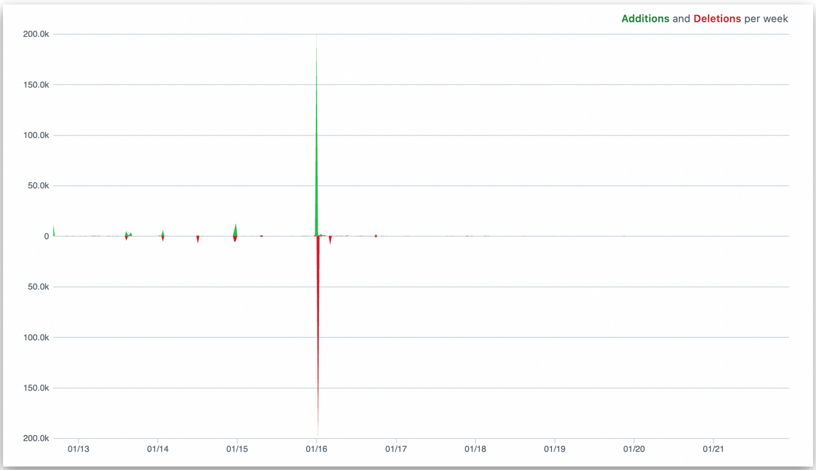


FIG. 1: CODE FREQUENCY OF THE **PYTHON-TWITTER** REPOSITORY.

three or fewer files; 46% of those were single-file commits. Interestingly, the **Python-Twitter** commits that affected more than nine files were often in the very high range, sometimes as many as 90 to 100 files per commit! The code frequency metrics show a high frequency of additions and deletions in early 2016 (see Fig. 1), where code updates and migrations may indicate a large number of files impacted in those particular commits.

If we consider Microsoft's definition of a "green" or good score for the **Maintainability Index (MI)**, we'll find that anything with a score of 20 or more is acceptable. Anything with a score of 10 to 19 is worth looking into (considered "yellow"), and nine or under is a "red" warning. In this regard, the **Requests** project has its lowest scored file at 20 MI, placing 100% of the **Requests** codebase in the good "green" category regarding the MI. **Fail2Ban** has six files that are currently below the "green" threshold, with three files in the "red" and three files in the "yellow" categories, accounting for 11% of the file count. **Python-Twitter** did not have many files for us to consider, but we saw only one file in the "red" category and 0 "yellow" files with what there were. Unfortunately, because of the low file count, this single file weighs heavily at 12.5% of the file count (more than in **Fail2Ban**, where six files accounted for both "red" and "yellow"!). Overall, **Python-Twitter** wasn't too bad off, with 87.5% of files considered "green" regarding maintainability.

Overall, it was interesting to review the various scores for each codebase. None of the repositories visited seem to be in a terrible place with overall maintainability via the metrics evaluated above, but there is still room for improvements. It is no easy task to keep an open-source project (or even a private project within a large company!) with great scores in every maintainability realm. The developers who have contributed to sustaining the integrity of the code through review and selective approval have done well so far!