# ⌨️ Choice of license in open-sourced Github repositories ⌨️

Author: Quan Phan

## 👋 Introduction

In software engineering, especially in the open-source world, a license of a project provides important information on how people other than the authors can use, distribute, and modify the project.

In this analysis, I want to understand how the choice of license changes across different open-sourced projects and communities. More specifically, I want to answer three questions...

1. What is the most popular license in the open-sourced world?
2. How does the choice of license differs for a popular, well-known project, compared to a regular-size open-source repo?
3. How does the choice of license differs across open-source communities (by programming language)?

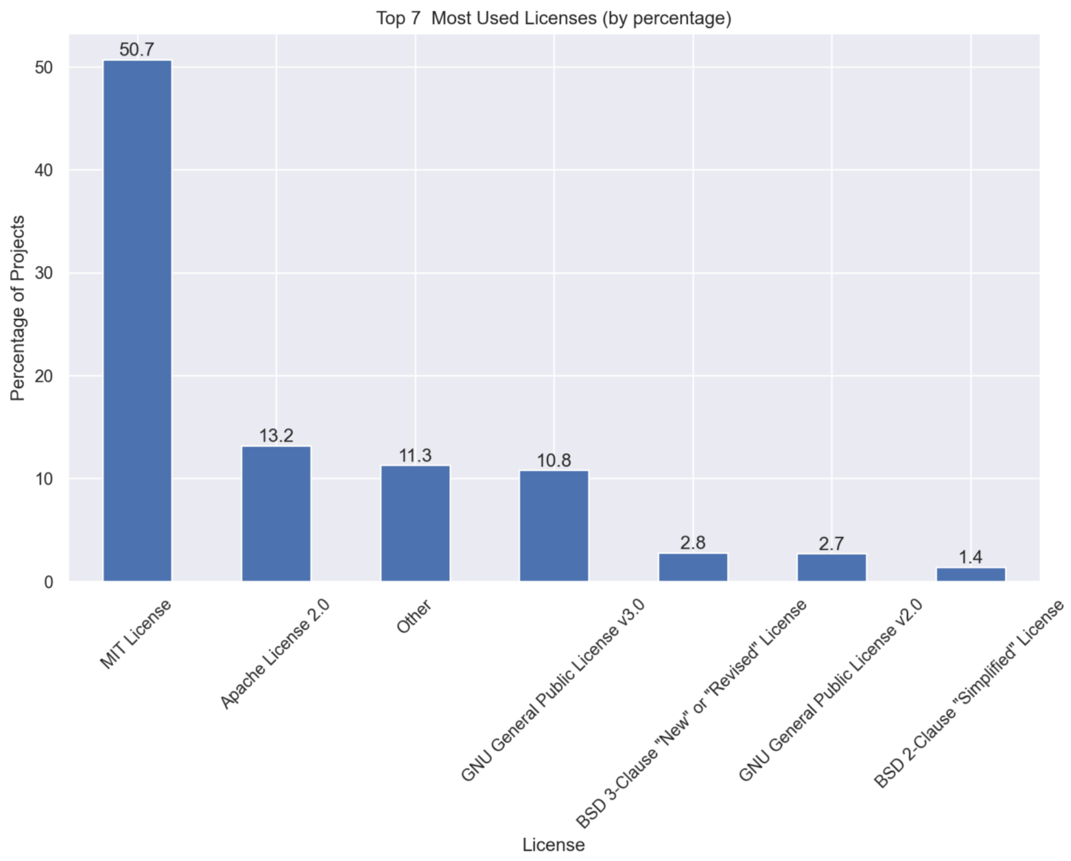
This analysis was inspired by my recent research and choice of license for a new open-source project.

## 🍌 Data

The dataset is a sample of 10,000 data points, taken from the version 2 of the Github Dataset on Kaggle. Readers can find the description of the data in the Kaggle link.

## 🏆 Top choice licenses

It is impressive to see 50% of the projects use the MIT license. The runner-ups are Apache License 2.0 (13.2%) and GNU GPL v3.0 (11.3%).



To understand the *why* behind the distribution of these three most popular licenses, let's learn about their most distinctive features:

**MIT license** is known for its simplicity and permissive nature. It allows users to do almost anything they want with the code, even distributing, sublicensing, and selling copies of the software. The only major requirement is to include the original copyright and license notice in any copy of the software/source code. Thus, it is not difficult to see why the MIT license, With its simplicity and minimal requirements, is the most appealing license for open-source developers.

**Apache License 2.0** is as permissive as MIT license, and users can do almost anything they want with the code. However, a distinctive feature of the Apache License is its explicit grant of patent rights from contributors to users, protecting users from patent litigation. In addition, it requires modifications to be documented, making it easier for subsequent users to understand what was altered.

In contrast to MIT and Apache 2.0, **GNU GPL v3.0** is known for its strong copyleft requirement. If you distribute modified versions of GPLv3-licensed software, you must also distribute the entire source code under GPLv3. This ensures that all modified versions of the software remain free and open.

In short, while being as permissive as the MIT license, both Apache 2.0 and GNU GPL v3.0 cannot top the champion due to their additional requirements, which significantly contribute to the maintenance 'costs' of the open-source project.

## 🍉 Prefered license types in popular vs. regular repositories

### Hypothesis

One hypothesis that might spring off from our understanding of the licenses is that...

- MIT license might be more appealing to a typical, small-scale project due to its simplicity.
- Conversely, the Apache 2.0 License is more appealing for larger, more popular, and commercially-oriented projects because it offers more protection against patent infringement claims and has explicit grants of patent rights.

## Popular vs. regular repositories

To examine this hypothesis, I need to distinguish large and popular repositories versus typical, small-scale ones. To do this, I use a proxy metric, that is the weighted average of a several numeric metrics in the dataset.

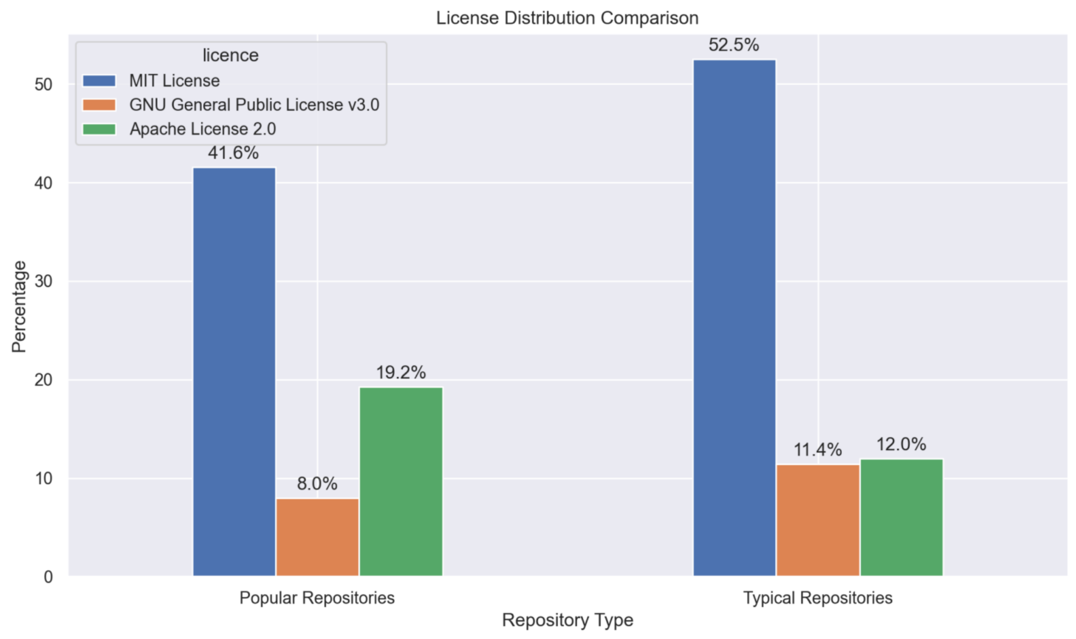
Here's my formal definition: A popular (and mature) project is any project with the **weighted average number of stars, watchers, forks, and pull requests larger than the upper interquartile range of this metric**. The rest, I call regular projects.

By this definition, we have 8787 typical repositories (87.87%), and 1213 popular repositories (12.13%).

In defining my proxy metric, I intentionally leave out the number of commits. My *assumption* is that the number of commits does not necessarily correlate with the maturity of a project. A high number of commits might just mean a high number of bug fixes. Alternatively, beginner developers can push many commits without good reasons. I also use the upperquartile range of this proxy metric as a cutoff point because as a norm in the Stats community, anything beyond the upperquartile range is considered outliers, i.e "the top of the top". In my analysis, I really want to focus on the constrast between this "extreme" group and the rest.

## Result

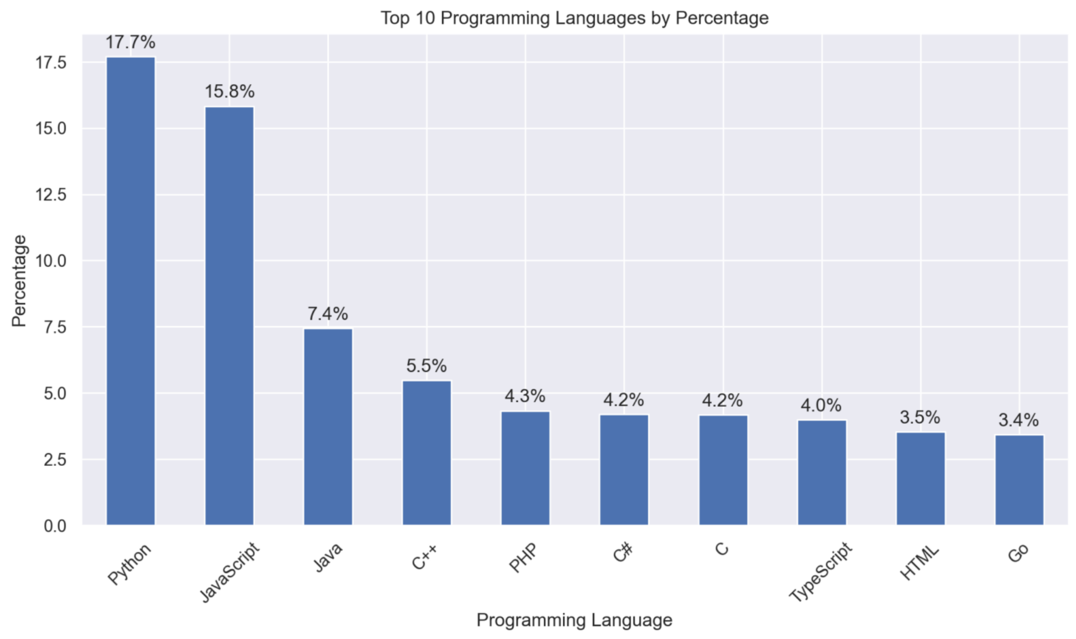
While MIT license dominates in both the popular and regular-size project groups, the percentage is considerably higher in the latter (10 percentage unit difference). Among the three, only Apache 2.0 has a higher usage rate in the popular project group compared to the typical group.



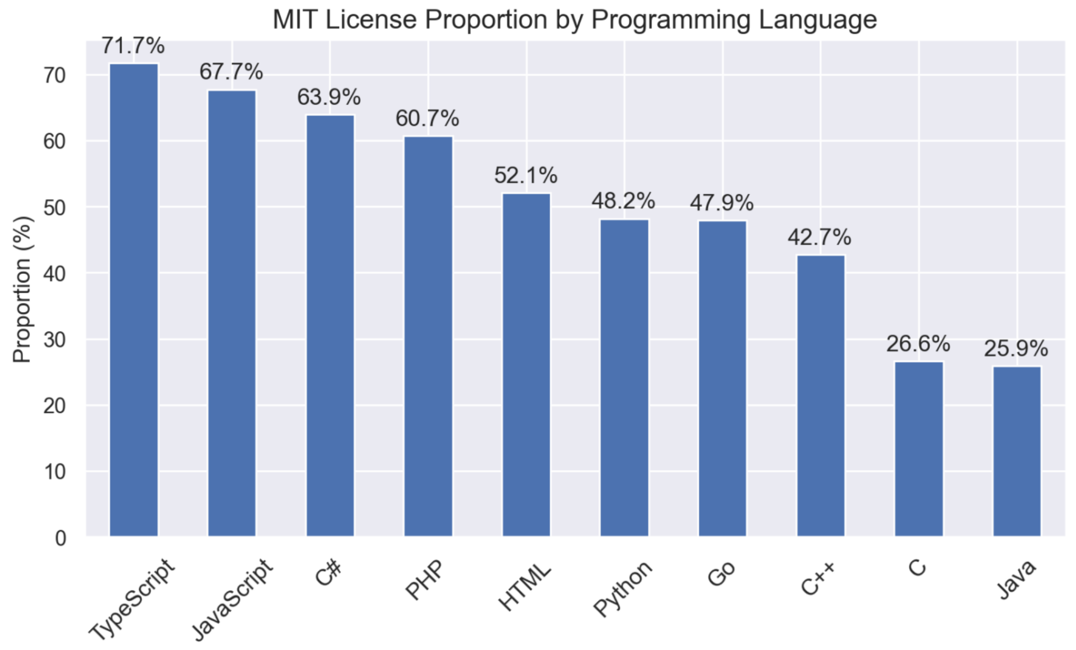
This result confirms the direction of my initial hypothesis, that a small-scale project is more likely to use the MIT license and the popular, mature projects are more likely to use Apache 2.0. However, the magnitude of difference is not as significant as I expected.

## ⌨️ Prefered license types across different open-sourced communities

In this section, I have no hypothesis to start out with, so let's begin by getting to know the largest programming communities:



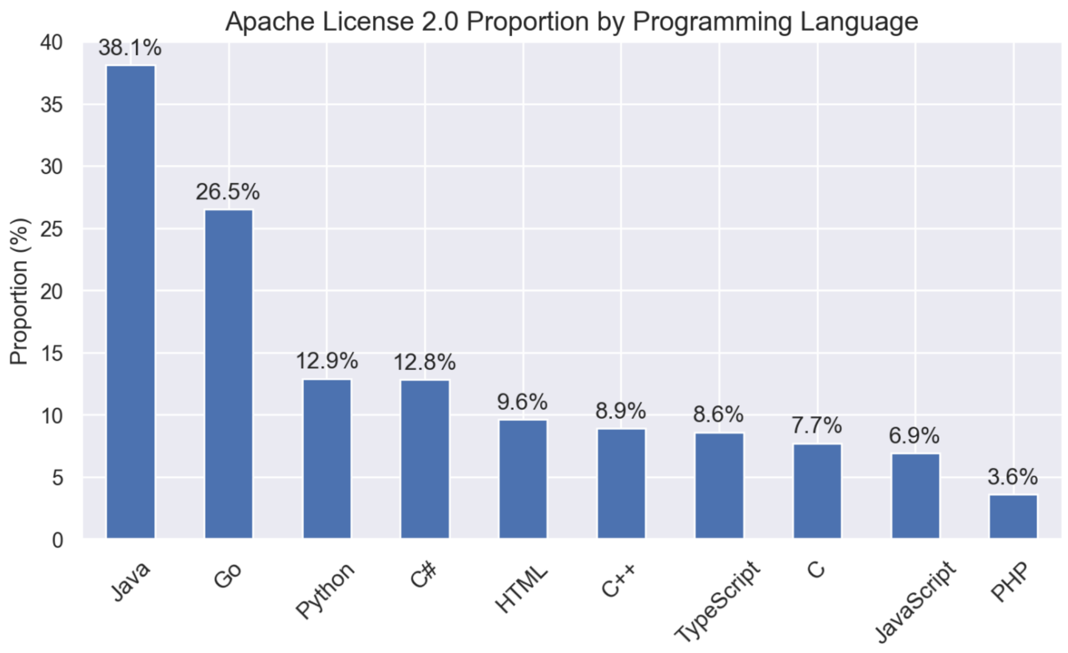
This is a well-expected list, with Python and Javascript taking the lead. Let's see how the choice of license differs across these programming-language communities.



The high usage proportion of MIT license in Javascript (JS) and Typescript (TS) projects (approximately 70% for both groups) well aligns with the strong culture of sharing and building upon others' work in the web development world (which heavily uses JS and TS).

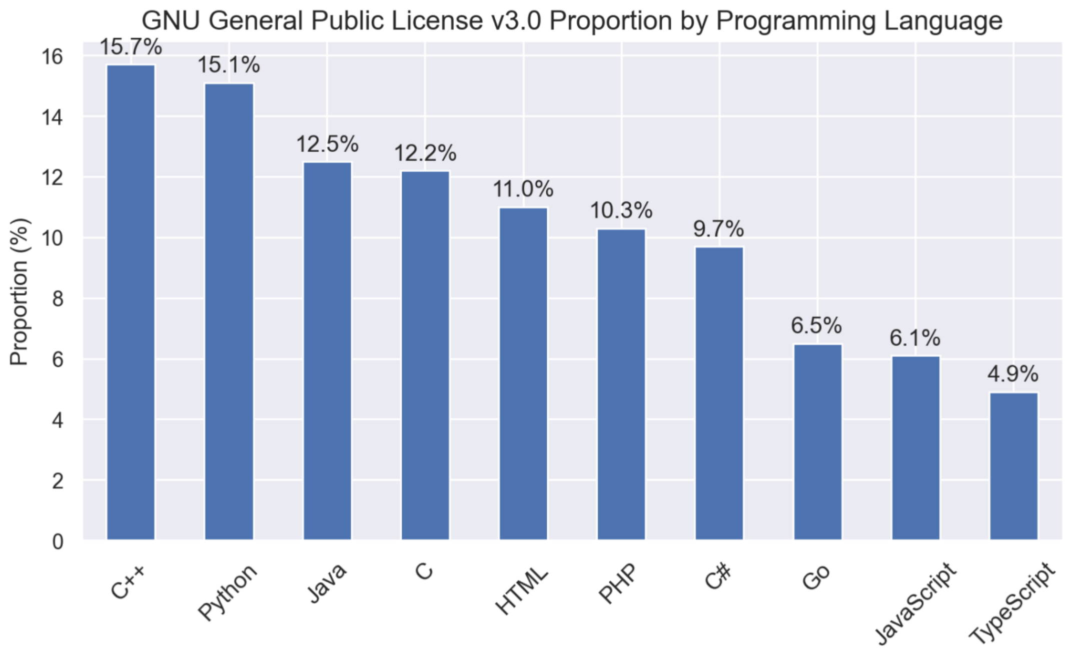
Another factor that might influence the wide adoption of the MIT license in the JS and TS communities is the fact that many large and influential projects in these languages (like Angular, React, Express.js) use the MIT License, setting a precedent for others in the community.

In contrast...



Apache License 2.0 is significantly more popular in the Java community than the rest. The connection between these two are more subtle. One hypothesis is that since both Java and Apache 2.0 (with its explicit grants of patent rights) are widely used in the enterprise environments, there are more Java projects with Apache 2.0 than other languages.

Finally...



GNU GPL v3.0 is more widely used in the C++ and Python communities than the rest, although the difference is not that significant.

## 🍊 Conclusion

In this analysis, I have explored and answered all the questions in the **Introduction** related to the choice of licenses. Here are some highlights...

1. The most popular license in the open-sourced world is MIT license, followed by Apache 2.0 and GNU GPL v3.0.
2. Compared to large, well-known project, a small-scale project is more likely to use the MIT license. Conversely, Apache 2.0 is more popular for large projects than small ones.
3. MIT license has the highest usage proportion in the web development community, while Apache 2.0 is the most popular in the Java community.

That's all for today. I hope you had fun following this analysis and it is somehow helpful for your choice of license in your next open-source project. Thanks for following along and will see you in an interview 😉