# Attributing Credit and Measuring Impact of Open Source Software Using Fractional Counting

**Anonymous**

## 1 Introduction

Open source software (OSS) is an innovation activity making the technology accessible and customizable for users around the globe. Widely used examples of OSS include Apache, Linux, R programming language, Mozilla Firefox. OSS is developed by a variety of entities, including universities, businesses, government research institutions, nonprofits, and individuals. While the developers of OSS are thought to be very widespread, there remains many questions to be answered about who these contributors are, who are the largest contributors (countries, sectors, organizations), and how they collaborate with each other.

This work is inspired by the National Science Board's Science & Engineering Indicators report that publishes trends and international comparisons for publications output (White, 2019). The report provides statistics on top contributing countries, international collaborations and citation patterns for published research articles. However, the creation and use of OSS software tools highlight an aspect of technology diffusion that is not captured in science and technology indicators.

In this paper, we focus on packages developed for programming languages R and Python and leverage methods developed in bibliometrics to measure the contribution of countries to OSS. In bibliometrics, attributing credit to publications' authors is typically done by giving each author an equal proportion of the credit, known as fractional counting (i.e., the "1/n" rule (de Mesnard, 2017)), in which each co-author of a work gets 1/n of the credit, where n represents the total amount of authors. The availability of data on OSS development (i.e., lines of code, contributors' affiliations and locations) yields an opportunity to attribute credit to authors more accurately. Using data collected on R and Python from GitHub, we measure the exact contribution of each developer (author) and use weighted counting based on the lines of code added by each developer to find top contributors to OSS. Moreover, using the dependency relationship between packages (similar to citations), we study the pairwise connections between countries to measure their respective impact.

## 2 Data and Methods

In this study, we focus on packages developed for R and Python. We employed web-scraping techniques to extract detailed package information from their respective package managers, CRAN for R and PyPI for Python. Most of these registries also contain metadata for the packages including information such as author and maintainer names, license, source code URL, and date of creation. Using source code URL's, we identified packages that are developed and maintained on GitHub and collect repository-level information from GitHub (including lines of code added and contributors) for the packages through their respective GitHub URL. Out of 19,852 R and 428,708 Python packages, we identified 7,844 R and 64,022 Python packages on GitHub. The data includes 14,328 unique R developers and 184,444 Python developers. We used the location information, email address, and organization information provided by the users to assign each user to a country. The country information was missing for 39% of the R developers, and 43% of Python developers.

### 2.1 Weighted Fractional Counting

Here we are focusing on a fractional counting method to attribute credit to countries contributing to OSS. However, packages have contributors from different countries, hence we leverage methods developed in bibliometrics, specifically the "1/n" rule (assuming equal contribution from each author), to attribute credit to authors. In our data, we have information about the lines of code added by each user, hence we can attribute credit based on the exact contribution of each user.

$$\frac{\text{loc}_i}{\sum_j \text{loc}_{ij}} \tag{1}$$

We use Equation (1) above to calculate the total lines of code (LOC) for a given repository, denoted as $\sum_j \text{loc}_{ij}$. It then divides each contributor's LOC (represented as $\text{loc}_i$) by this total. These individual contributions are aggregated based on the countries of interest.

### 2.2 Fractional Dependency Graph

One method to measure impact and diffusion in bibliometrics is to use citations to analyze affinity between entities like countries and universities. The National Science Board's Science & Engineering Indicators report publishes trends and international comparisons for publications output (White, 2019). They calculate the fractional citations between each pair of countries by taking the fractioned counts from both the citing article and the cited article at the author level using the "1/n" rule. They sum these fractions to calculate the fractional citation between each pair of countries (Science-Metrix, 2018). We adopt this method and apply it to packages by using the dependency/reuse relationship similar to citations.

Take the example of two packages A and B where package A reuses/cites package B. Say package A has two authors, one from the United Kingdom and one from China; and Package B

has two authors, one from the United Kingdom and one from the United States. Login 1 and login 2 are responsible for 80% and 20% of the lines of code in package A, respectively, and login3 and login 4 are responsible for 75% and 25% of the code in package B, respectively. For one dependency relationship (as in this example), the dependency fraction between each unique pair of countries is calculated by multiplying the respective package fractions.

| Citing login | Citing country | Citing package fraction | Cited login | Cited country | Citing package fraction | Dependency fraction |
|---|---|---|---|---|---|---|
| login1 | U.K. | 0.8 | login3 | U.S. | 0.75 | 0.6 |
| login1 | U.K. | 0.8 | login4 | U.K. | 0.25 | 0.2 |
| login2 | China | 0.2 | login3 | U.S. | 0.75 | 0.15 |
| login2 | China | 0.2 | login4 | U.K. | 0.25 | 0.05 |

Table 1: Fractional dependency counts between two packages

Finally, we use the sum of the dependency fractions for each unique pair of countries to develop the fractional dependency graph. In this graph, a link from country $i$ to country $j$ indicates that $i$ cites/reuses packages of $j$ and the weight of the edge corresponds to the sum of the dependency fractions from $i$ to $j$.

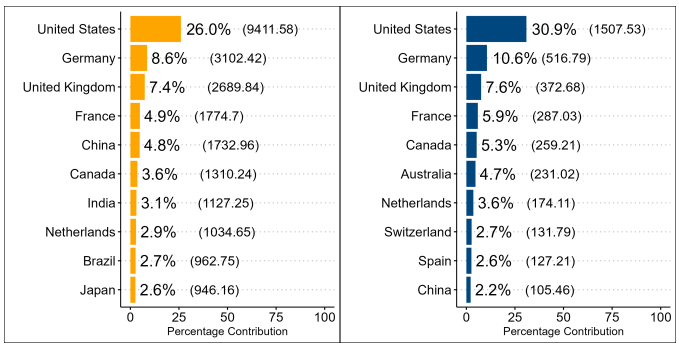
## 3 Results

### 3.1 Weighted contributions by country

Figure 1 illustrates the top contributing countries to R and Python calculated using the weighted fractions based on lines of code added by each user.

We find that for Python, the United States leads with a significant margin followed by Germany, United Kingdom, France, and China. United States accounts for 9,411.58 repositories corresponding to 26.0% of all Python repositories (excluding the contributions from users with unknown countries). Similarly, United States, Germany, United Kingdom and France are the top contributors for R. These top 10 countries make up of the 76.1% of the total R repositories, and 66.6% of Python repositories.

Figure 1: Top countries contributing to R and Python




### 3.2 Fractional dependency among countries

We develop the fractional dependency graphs (as described in Section 2.2.) and obtain a network with 80 nodes (countries), and 937 edges (dependency pairs) for R.[1] For illustrative purposes, Figure 2 shows the connections between the top countries based on their indegree (i.e., countries with the most highly reused packages). The size of the node represents the indegree, the larger the node, the higher the sum of overall dependency fractions for the country is. United States has the highest indegree, followed by Germany, Denmark, France, Norway. United States also has the highest outdegree (number of packages reused from other countries), followed by Germany, United Kingdom, and Australia.

Figure 2: Fractional dependency links between top countries


,

When we analyze the pairwise dependency fractions, we observe that United States' packages are most frequently reused by packages from Germany, Spain, Italy, Australia, and United Kingdom based on the total dependency fractions. In parallel, United States mostly uses packages from Germany, France, and Denmark. We also see that countries tend to either cite/reuse the United States packages or packages from their own country.

## 4 Discussion

It should be noted that our results do come with some shortcomings. First, not all R and Python packages are developed on GitHub. Furthermore, we rely on self-reported data in GitHub profiles to assign countries to users which leads to missingness and potential inaccuracies. Aside from these shortcomings, these results give motivation to work with much larger samples. The analysis only focused on packages from R and Python, but this method could be expanded to all repositories on GitHub, and collecting forks could serve as a way to analyze impact through reuses. This framework could also be applied to measure contributions from various sectors, organization and institutions.

## References


Louis de Mesnard. 2017. Attributing credit to coauthors in academic publishing: The 1/n rule, parallelization, and team bonuses. *European Journal of Operational Research*, 260(2):778–788.

Science-Metrix. 2018. Bibliometrics and Patent Indicators for the Science and Engineering Indicators. Technical Doc.

Karen White. 2019. Publications output: U.S. trends and international comparisons. Science & Engineering Indicators 2020. NSB-2020-6. *National Science Foundation*.


[1]The users with unknown countries are removed from the network.