TDS September 2024 Project 1: Summary

Contents

1.Introduction

In this repository, we present the study of Chicago-based GitHub users who possess over 100 followers. This study looks at their profiles and publicly available repositories. The purpose of this study is to examine their activities, contributions, and skills by systematically organizing the information by utilizing the GitHub API.

The data has been logically arranged in two files: users.csv - This file describes the profiles of users, which cover:

- Username and Full Name: The names used to identify each user.
- Company: The cleaned company names where they work. No leading symbols, no excessive whitespaces and converted in upper case.
- Location: Chicago.
- Bio: Provided to us by the user, a small piece of information about him or her.
- Public Repos, Followers, Following: Information about each user's contribution, public as well as follower network.
- Hireable and Email: These two columns depict if they are seeking employment and contact details if provided.
- Created Date

Rerepositories.csv – this file includes the details for each user in users.csv up to 500 most recent public repositories.

- Repository Name: Complete name with the username last (i.e. johndoe/project1).
- Created Date, Language, Stars, Watchers: Date when the repository was created, the language used, and the popularity of the repository, respectively.
- Features and License: Which features work such as who can edit, project boards and wikis et cetera, and what kind of license for the repository.

In making this data available in the form of CSV helps in identifying aspects of the Chicago GitHub community such as popular technologies, networking and leading experts etc.

2.Approach

The following steps are performed from the collection of data down to uploading the final files to a GitHub repository:

Step 1: Identify and select appropriate Tools

Different Tools and programming languages can be used. In this project Python and related libraries are uses

Tools & Soaftware Programs : Python,Panda Google Colab, Excel etc.

Step 2: Create GitHub Repository

Create a new GitHub repository, say TDS Project1, where the project files will be stored.

It should be initialized with a README for documentation purposes.

Step 3: Collect User Data from GitHub API

Retrieve users in Chicago with over 100 followers using GitHub's REST API. For Authentication create a GitHub API token to prevent rate limits and allow for pagination if one is dealing with large amounts of data.

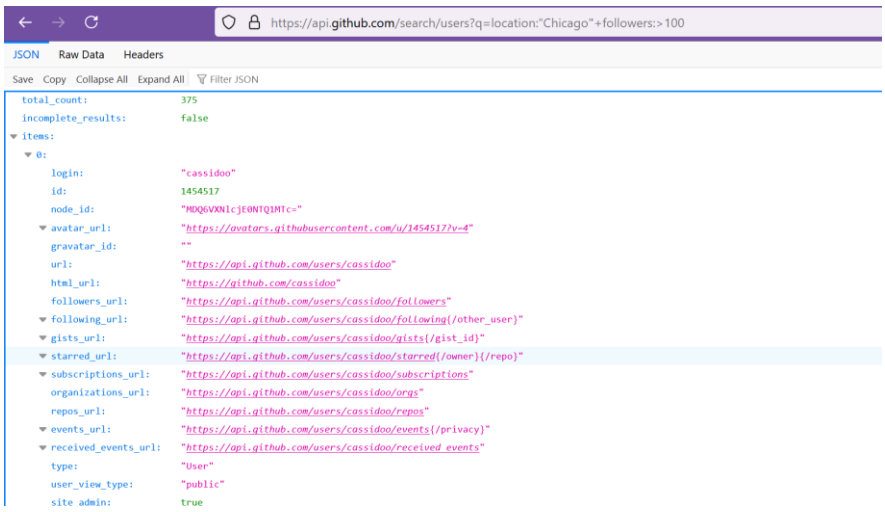
For setting up PAT(personal access token) refer the following link

https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens

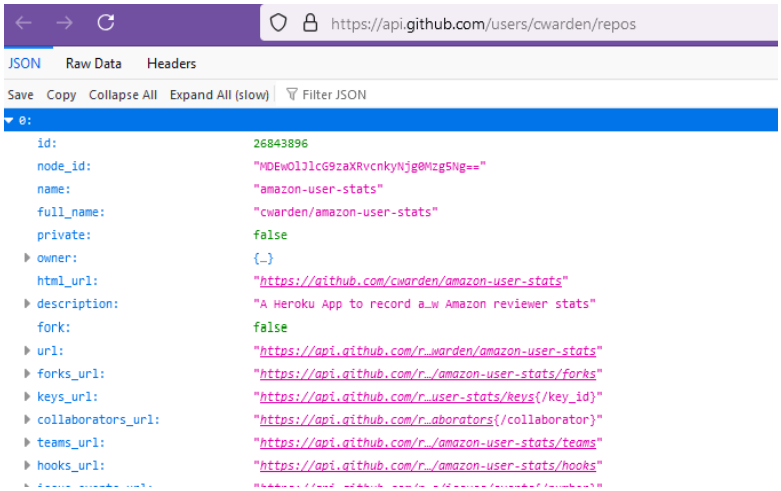
Endpoint: Utilize the /search/users endpoint with parameters such as q=location:Chicago followers: >100 to filter and search for users in Chicago that have over 100 followers.

See the following example how to access data using api end point

https://api.github.com/search/users?q=location:%22Chicago%22+followers:%3E100



https://api.github.com/users/cwarden/repos



For each user, gather information using the endpoint /users/{username}, fetch fields: login, name, company, location, email, hireable, bio, public_repos, followers, following, created_at.

Save all this information to a file called users.csv, formatting properly, for example, converting boolean data into true and false, replacing nulls with empty strings, clean up the name of companies.

Step 4: Get Repository Information of Each User

Fetch the list of public repositories for every user in users.csv using the /users/{username}/repos endpoint. To do so, pagination should be applied to collect the 500 most recently updated repositories for each user. Extract and retain in repositories.csv the following fields: login full_name created_at stargazers_count watchers_count language has_projects has_wiki license.name Ensure the coherence of all data coming from doing the following:

Convert booleans to true and false. Fill in missing fields as empty strings. Store each repository associated with its user's login.

Step 5: Clean the Data and Format It

Company field cleaning in users.csv:

Strip leading @ symbols, trim whitespace and convert names to uppercase.

Validate CSV files by reviewing a sample to make sure all the required fields have been correctly filled and formatted.

Make use data field holding Boolean is converted true-false and null value false

Step 6: Create Documentation

Create a README.md file detailing:

Approach, Insights and any action points.

Describe the data in users.csv and repositories.csv.

Any supporting code and/or analysis artifacts if you chose to support the deliverable with additional components.

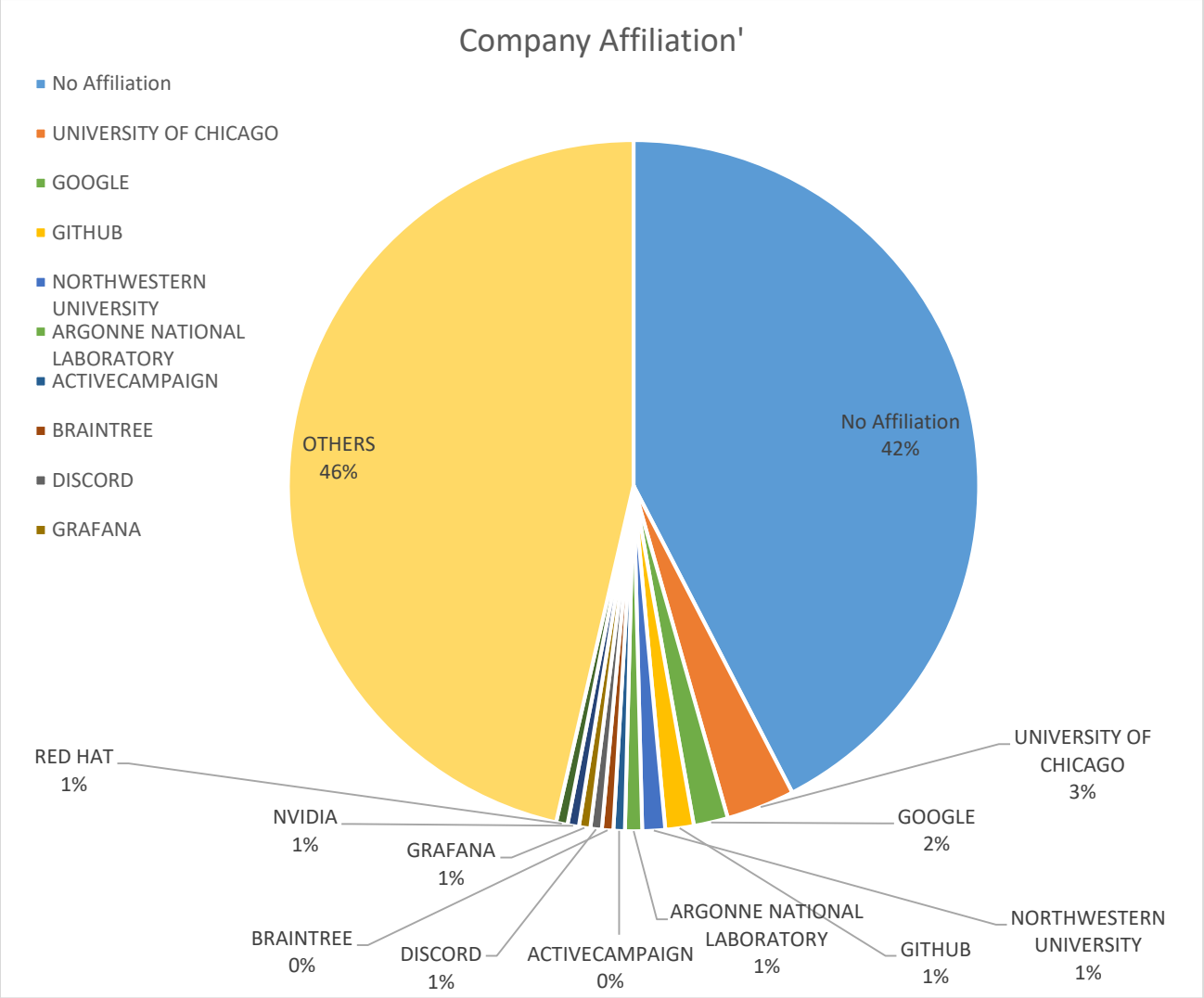
Step 6: Upload Files to GitHub

Commit and push users.csv, repositories.csv, and README.md to the main branch of the Github repository.

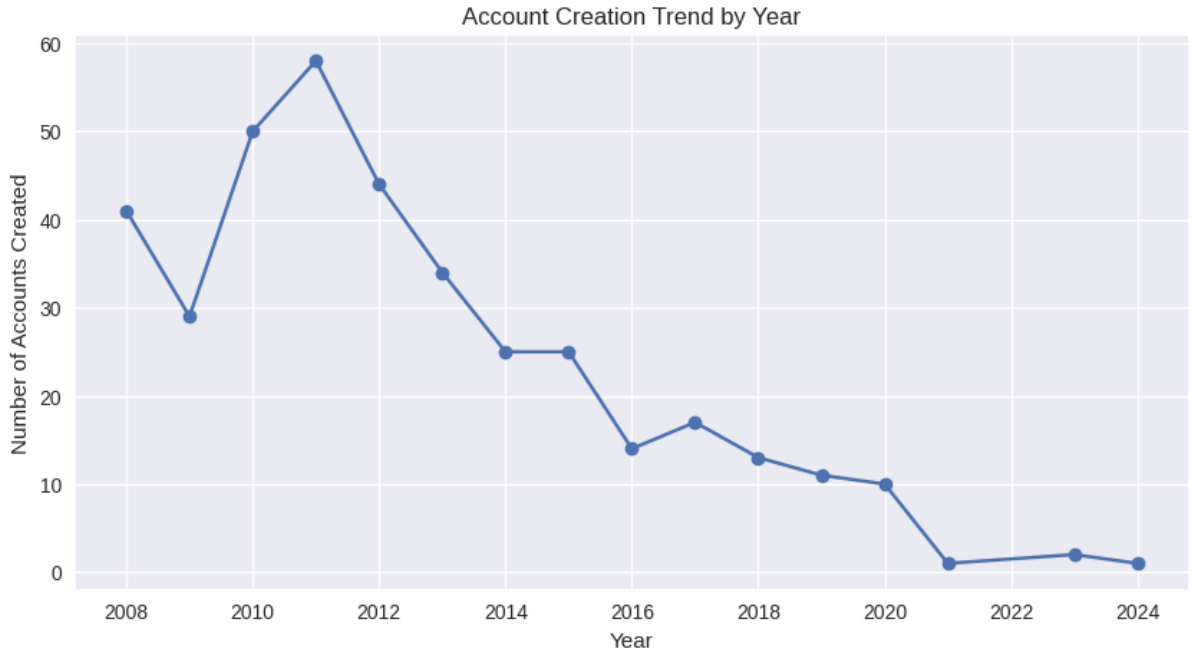
3.Analysis

1. Users & company affiliation

Users with more than 100 followers as of 26th October 2024	375
No Affiliation	159
UNIVERSITY OF CHICAGO	12
GOOGLE	6
GITHUB	5
NORTHWESTERN UNIVERSITY	4
ARGONNE NATIONAL LABORATORY	3
ACTIVECAMPAIGN	2
BRAINTREE	2
DISCORD	2
GRAFANA	2
NVIDIA	2
RED HAT	2
OTHERS	174
	375


Company Affiliation'
No Affiliation
UNIVERSITY OF CHICAGO
GOOGLE
GITHUB
NORTHWESTERN UNIVERSITY
ARGONNE NATIONAL LABORATORY
ACTIVECAMPAIGN
BRAINTREE
DISCORD
GRAFANA
No Affiliation 42%
OTHERS 46%
UNIVERSITY OF CHICAGO 3%
GOOGLE 2%
NORTHWESTERN UNIVERSITY 1%
GITHUB 1%
ARGONNE NATIONAL LABORATORY 1%
ACTIVECAMPAIGN 0%
DISCORD 1%
BRAINTREE 0%
GRAFANA 1%
NVIDIA 1%
RED HAT 1%

2. Users account creation trend



This graph shows the trend in GitHub account creation over time from 2008 to 2024. Here are the key observations:

1. Peak Period:
   - Account creation reached its peak around 2011-2012 with nearly 60 new accounts
   - There was a notable increase from 2010 to 2011, showing rapid platform adoption
2. Decline Phase:
   - Sharp decline from 2012 to 2016
   - More gradual but consistent decrease from 2016 to 2022
   - Numbers dropped from ~55 accounts in 2011 to less than 5 by 2022
3. Recent Years (2022-2024):
   - Very low account creation rates
   - Slight fluctuation but staying under 5 new accounts per year
   - Appears to have stabilized at this lower level
4. Overall Trend:
   - Clear downward trend over the 16-year period
   - Three distinct phases:
     - Growth phase (2008-2011)
     - Rapid decline (2012-2016)
     - Gradual tapering (2016-2024)

This pattern suggests:

- GitHub had its highest growth in new users during its early years
- The platform may have reached market saturation

- Most developers who need GitHub accounts likely already have them
- The current low rate might represent normal replacement or new entry into the field

Note: This data might represent a specific subset of accounts under Chicago rather than all GitHub accounts, as the total GitHub user base is known to be much larger.

3. Top 10 users with highest followers.

company	public_repos	followers	following	created_at	days_since_creation	hireable
GITHUB	165	13367	102	20/02/2012 16:36	4632	FALSE
SHOREBIRDTECH	125	8669	67	22/09/2014 02:35	3687	FALSE
DABEAZ, LLC	34	5175	0	01/08/2010 15:22	5200	FALSE
	24	3762	0	08/03/2008 22:17	6075	FALSE
AQUATIC CAPITAL MANAGEMENT	84	3396	97	23/02/2011 13:46	4994	FALSE
GOOGLE	229	2729	259	18/01/2018 19:36	2472	FALSE
NIELSEN	104	2663	0	13/06/2012 23:23	4517	TRUE
	6	2451	0	20/07/2017 16:27	2655	FALSE
NVIDIA	143	2303	0	24/02/2008 18:54	6089	FALSE
	471	1849	44	16/03/2011 13:44	4973	TRUE

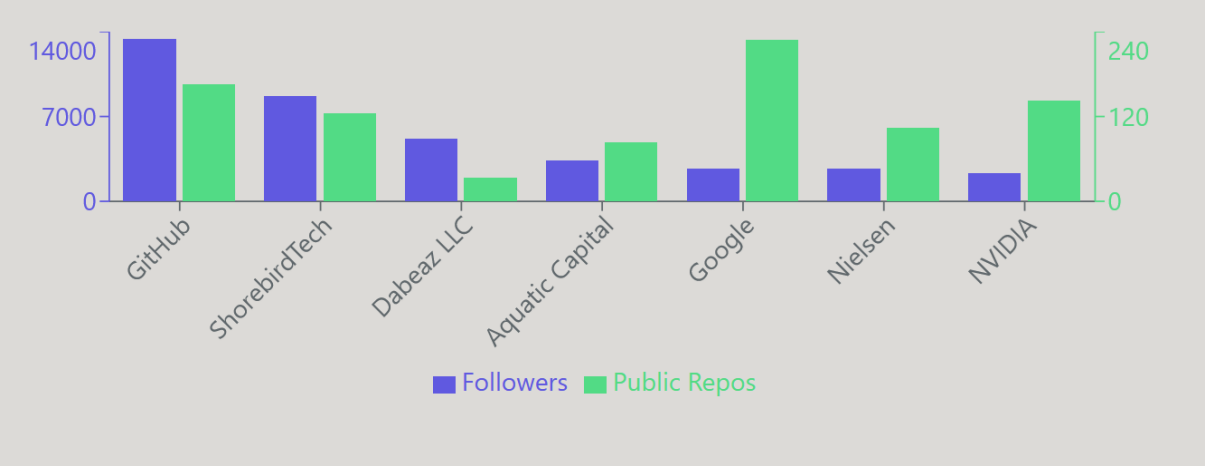
Most of the users with a large number of followers are working with major technologies and most of them created a user account 10 years back. Also, note that the majority of them are not following fewer people. Hence, users having a high count of followers working for major tech companies or having large public visibility implies that they have recognition as experts. Their huge followings are indicative that many people look up to them for knowledge or insight into their respective fields. Hence, most of these accounts with high levels of followership were all created a decade ago, which is actually ample time to gather any number of followers. In most cases, longevity on the platform usually means credibility because long-standing accounts most probably signal continued contributions or expertise in technology and development.

Many of the high-follower accounts are not hireable, which could mean that they are well established already in their current roles or chose not to indicate their availability for work. This also befits their status as experts, in that they may already hold prestigious positions within organizations or fields.

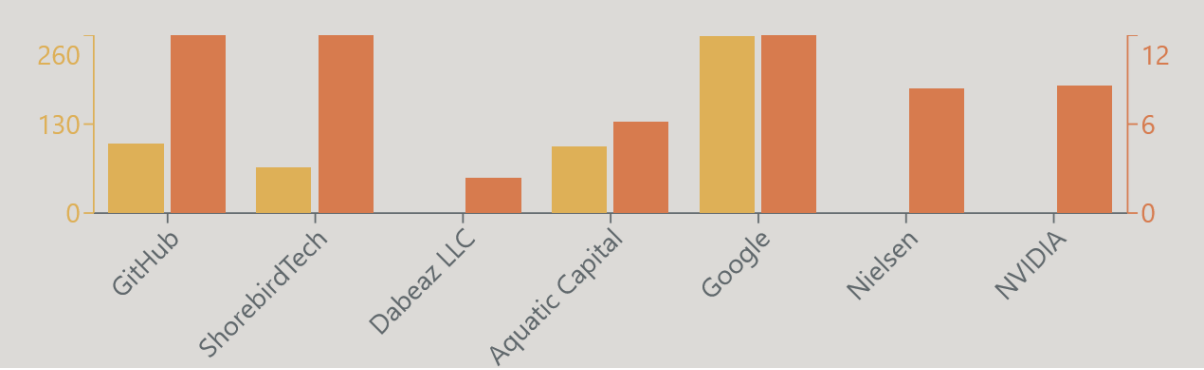
4. User Engagement Metrics by Top 7 Company Affiliation

Company	Followers/Following Ratio	Repos/Year	Account Age (Days)
GITHUB	131.05	13	4,632
SHOREBIRDTECH	129.39	12.37	3,687
DABEAZ, LLC	N/A	2.39	5,200
AQUATIC CAPITAL	35.01	6.14	4,994
GOOGLE	10.54	33.81	2,472
NIELSEN	N/A	8.4	4,517
NVIDIA	N/A	8.57	6,089

Followers and Repository Count by Company Affiliation



User Account Count by Company Affiliation



Based on the data, here's a comprehensive assessment of GitHub users by company affiliation:

1. Influence and Reach:

- GitHub-affiliated user has the highest follower count (13,367), suggesting significant influence in the community
- ShorebirdTech-affiliated user shows strong community presence with 8,669 followers
- Users from traditional tech companies (Google, NVIDIA) have moderate follower counts (2,000-3,000)

2. Engagement Patterns:

- Google-affiliated user shows highest bilateral engagement (259 following, 2,729 followers)
- Users from DABEAZ, Nielsen, and NVIDIA don't follow others, suggesting a one-way engagement model
- GitHub and ShorebirdTech users maintain moderate following counts relative to their follower base

3. Repository Activity:

- Google-affiliated user is most active with 229 public repos despite having a newer account
- Activity levels vary significantly across companies, not necessarily correlating with company size
- NVIDIA-affiliated user shows consistent long-term activity (143 repos over 6,089 days)

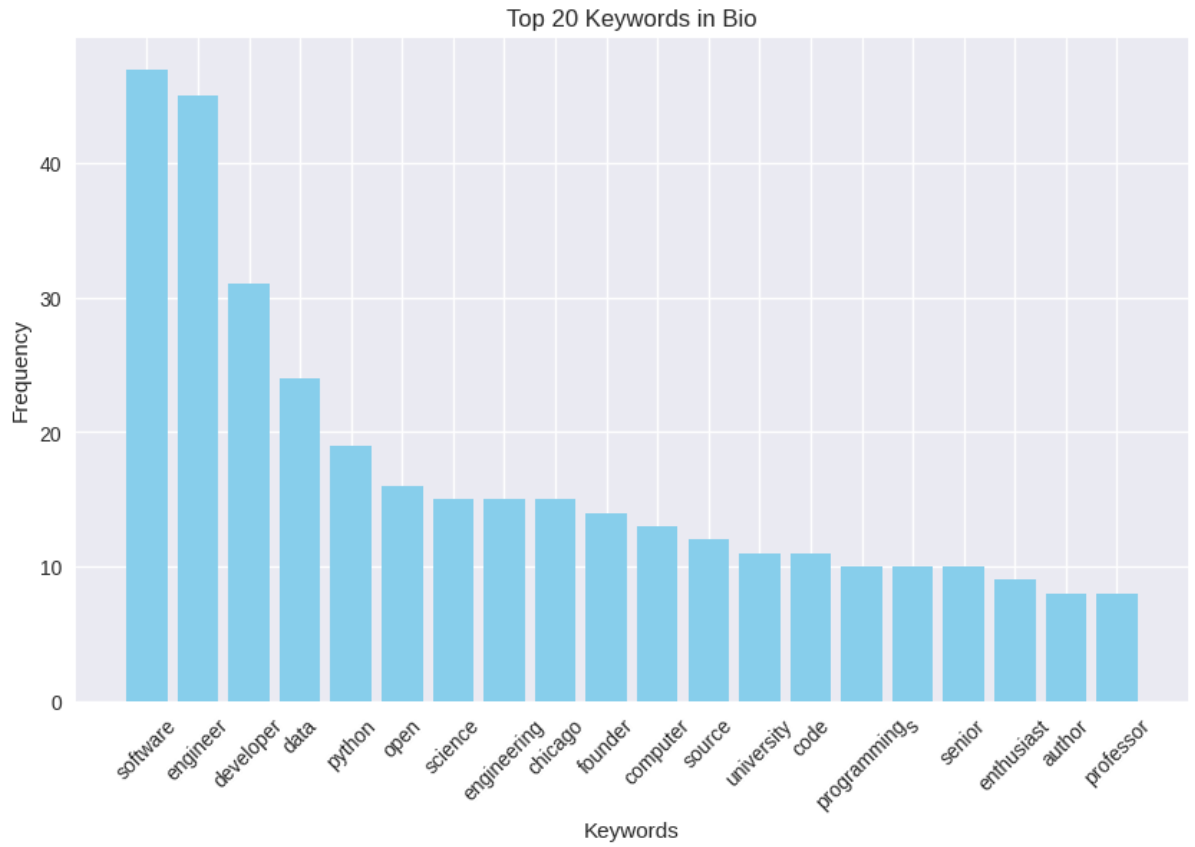
4. Account Maturity:

- NVIDIA user has the oldest account (since 2008)
- Google user has the newest account (since 2018)
- Most users established their accounts between 2010-2014

5. Engagement Style by Company Type:

- Tech Platform Companies (GitHub, ShorebirdTech): High follower counts, moderate following
- Traditional Tech Companies (Google, NVIDIA): Moderate followers, varying engagement styles
- Specialized Companies (DABEAZ, Aquatic Capital): Mixed patterns, suggesting individual-driven engagement

5. Top 20 key word used in Bio.



Key Observations:

6. Top Keywords:

- "software" appears to be the most frequent term with ~45 occurrences
- "engineer" is a close second with ~43 occurrences
- "developer" is the third most common with ~30 occurrences

7. Grouping Patterns:

- Technical Role Terms: software, engineer, developer, engineering, programmer
- Technology Fields: python, science, tech
- Professional Skills: passionate, research, creative

8. Distribution Pattern:

- There's a steep drop from the top 3 keywords to the rest
- The middle range (keywords ranked 4-12) shows a gradual decline
- The bottom range (keywords ranked 13-20) plateaus with similar frequencies around 8-10 occurrences

9. Professional Context:

- The keywords strongly suggest this is from analyzing bios of technology professionals
- There's a mix of both technical skills and soft skills/interests

- The prevalence of "software" and "engineer" suggests many bios are from software engineers

From this, it could be inferred that the dataset probably represents a community of professionals experienced in technology, software engineering, and data science, and a large number involved with open-source projects or academia. Professional roles assigned to them, leadership positions within communities, and interest in sharing their knowledge either through open source or academic writing suggest that they are influential personalities or thought leaders in their respective domains. This group also combines experience in industry practice and academic theory, placing them in an avid position within the tech community.

6. Correlation between followers, following and public repos

Correlations:			
	followers	following	public_repos
followers	1	0.032839	0.082159
following	0.032839	1	0.129014
public_repos	0.082159	0.129014	1

Followers and Following 0.032839

This is a very small correlation that approaches 0, which means that there is almost no relation between the number of followers a user has to the number of people they follow. This says that a user having a lot or a little amount of followers is irrelevant to the amount of accounts they follow.

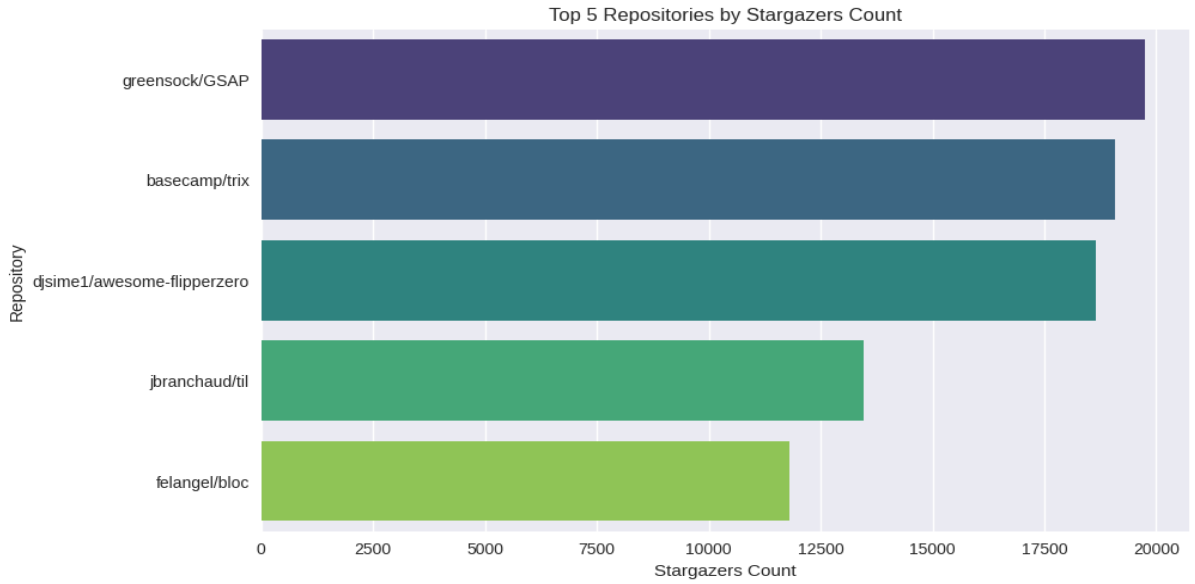
Followers and Public Repos 0.082159

This tiny positive correlation would imply the relationship between the number of followers a user has and the number of public repositories is weak. Users who have more repositories will be inclined to have a few more followers, but this relation is very weak; thus, other factors most likely influence the number of followers more strongly.

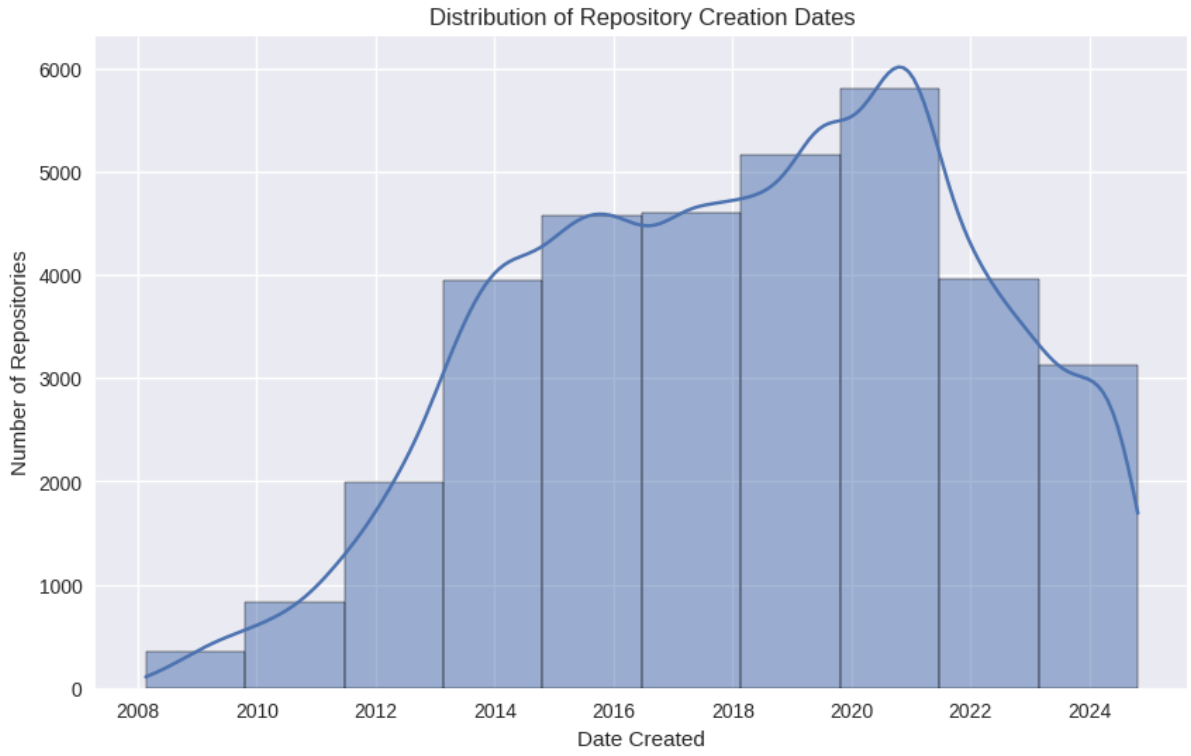
Following and Public Repos: 0.129014

A similar situation with a weak positive can be observed here, which shows that users with more repositories slightly tend to follow more users. In this case, though, the value of the correlation is very low, indicating that the number of repositories that a user has does not strongly influence how many people he or she follows.

7. Top Repositories by Stargazers Count:

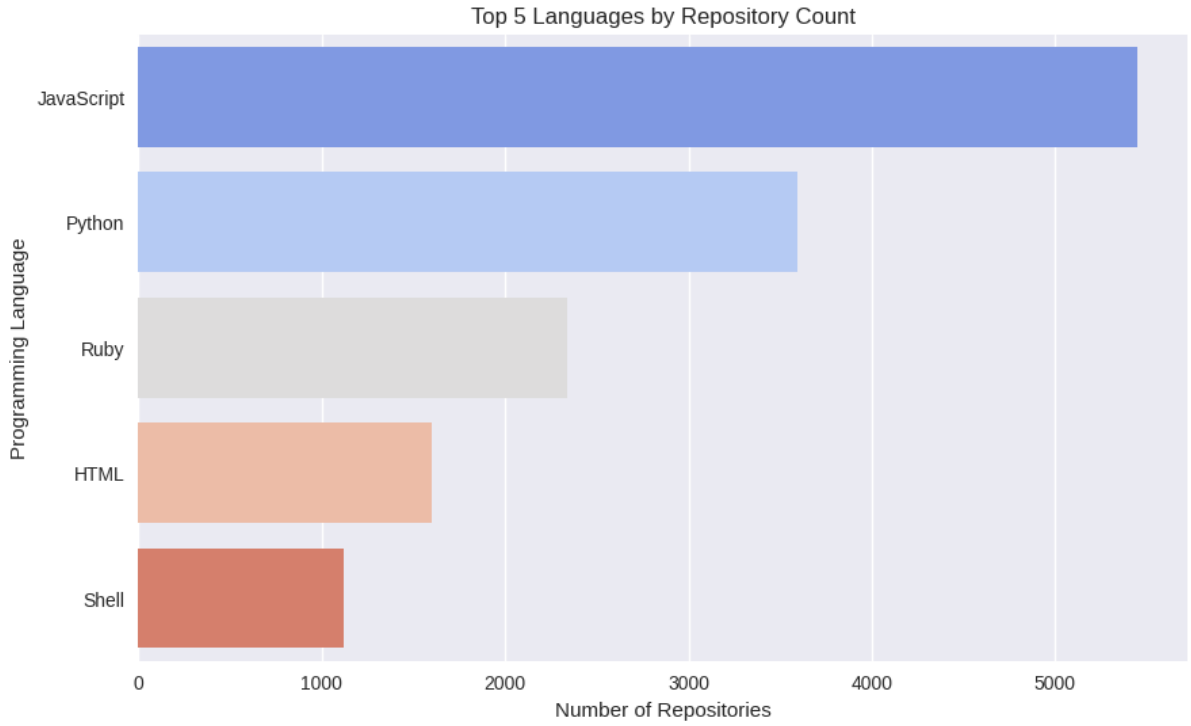


8. Distribution of Repository Creation Dates

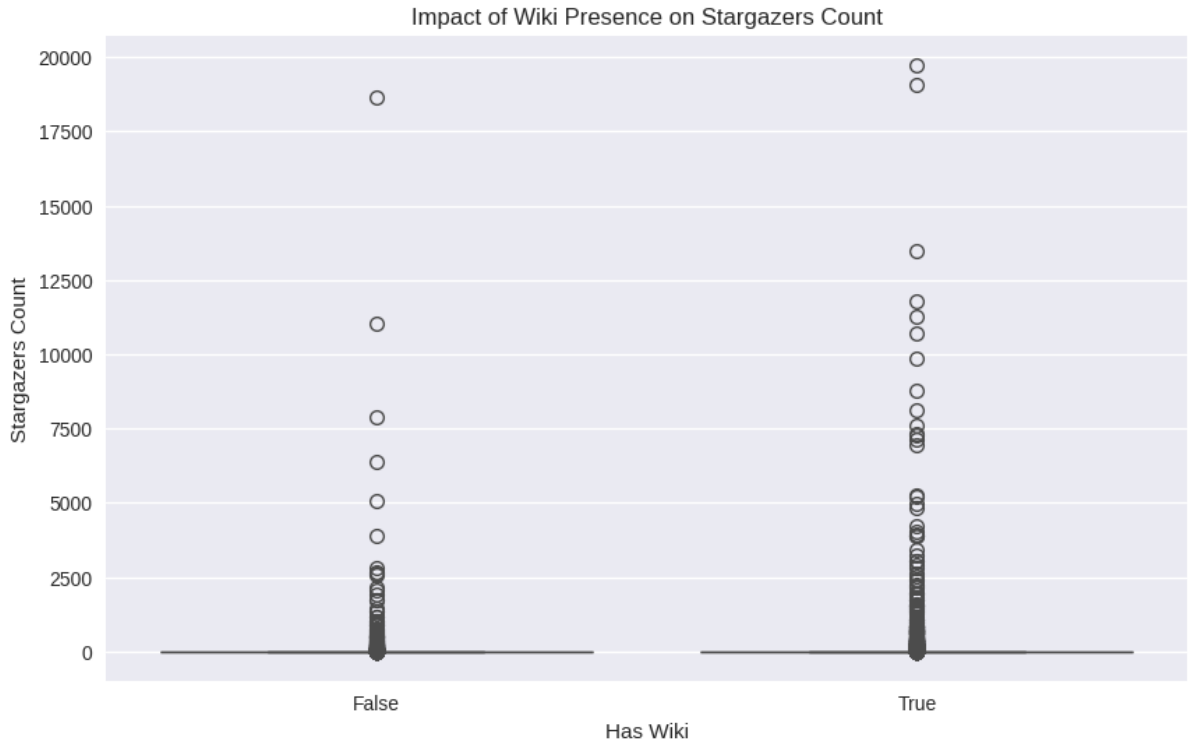


9. Top5 languages by number of repositories

Top 5 languages by number of repositories	
JavaScript	5452
Python	3597
Ruby	2339
HTML	1601
Shell	1117



10. Impact of Wiki Presence on Stargazers Count



This scatter plot shows the relationship between a repository having a wiki and its number of stargazers. Here are the key observations:

10. Distribution Pattern:
    - Both repositories with and without wikis show a similar "long tail" distribution
    - Most repositories cluster at lower stargazer counts (below 5,000)
    - A few outliers reach very high stargazer counts (up to ~20,000)

11. Repositories With Wikis (True):
    - Show slightly higher concentration in the middle range (5,000-15,000 stars)
    - Have more visible outliers in the high stargazer range
    - Appear to have more repositories with moderate stargazer counts

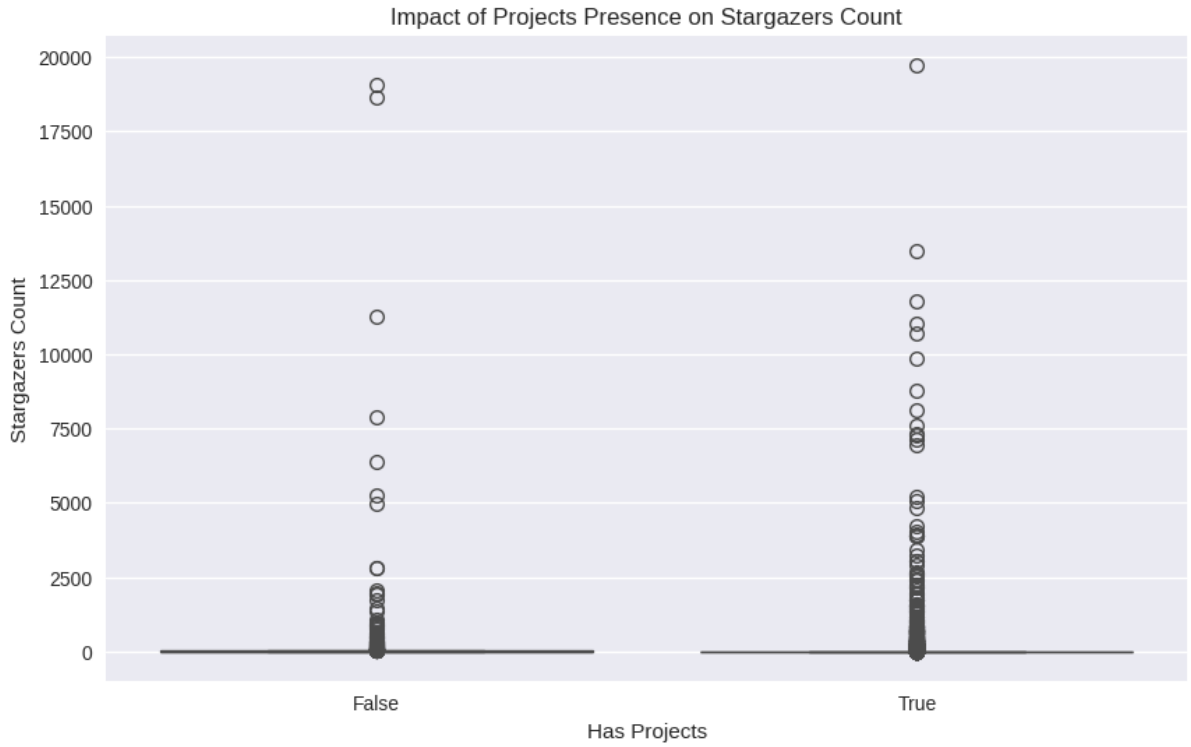
12. Repositories Without Wikis (False):
    - Also have some high-performing outliers
    - Generally show lower concentration in middle ranges
    - Majority cluster at lower stargazer counts

13. Overall Impact:
    - Having a wiki doesn't guarantee more stargazers
    - However, there appears to be a slight positive correlation
    - More popular repositories (>5,000 stars) seem more likely to have wikis

14. Insights:
    - The relationship between wikis and stargazers might be bidirectional:
        - Popular projects might be more likely to create wikis
        - Projects with good documentation (wikis) might attract more stars
    - The presence of outliers in both categories suggests that wikis alone don't determine a project's popularity

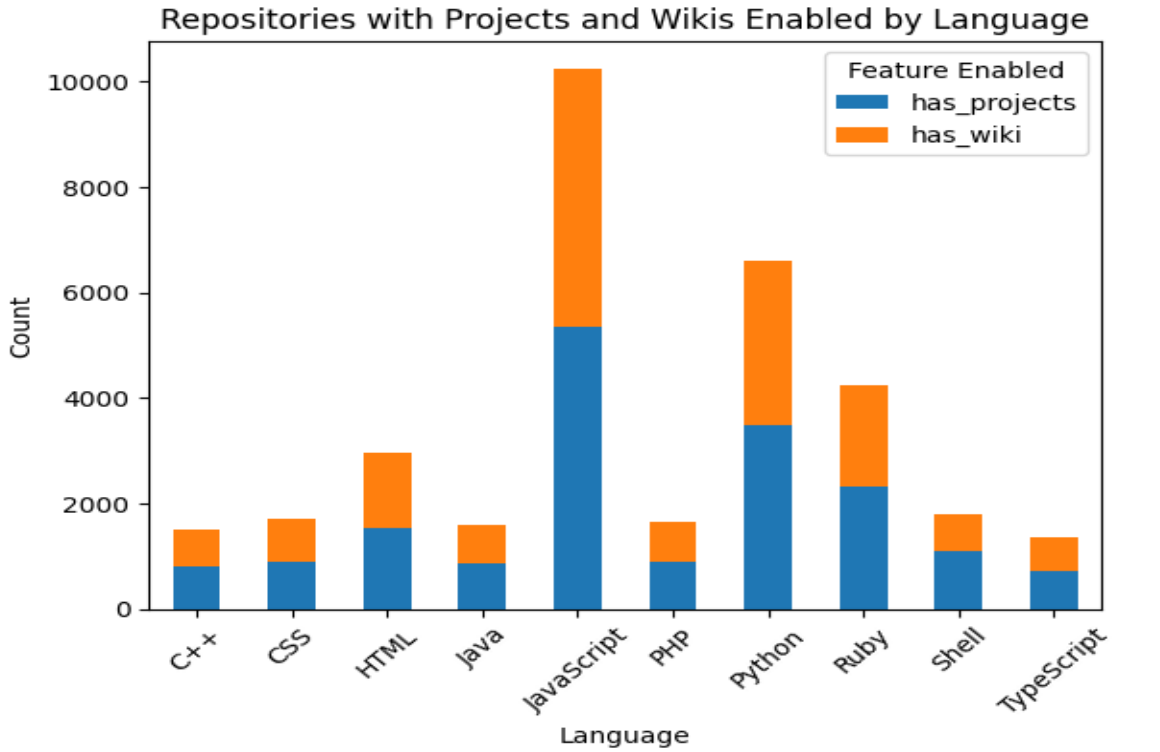
This suggests that while wikis may contribute to a repository's success, they are just one of many factors influencing a project's popularity on GitHub.

11. Impact of Project Presence on Stargazers Count



This shows that repositories with more project attract more attention.

12. Repositories with Project and Wiki Enabled by top 10 language



This stacked bar graph shows the distribution of repositories with Projects and Wikis enabled across different programming languages. Here are the key observations:

1. JavaScript dominates overall:
    - Highest total count at around 10,000 repositories
    - Roughly even split between Projects and Wikis
    - Shows the highest adoption of both features

2. Python is second most popular:
    - Around 6,500 total repositories
    - Higher proportion of Wikis compared to Projects
    - Strong documentation culture reflected in Wiki usage

3. Ruby comes in third:
    - Approximately 4,200 repositories
    - Also shows preference for Wikis over Projects
    - Consistent with Ruby's emphasis on documentation

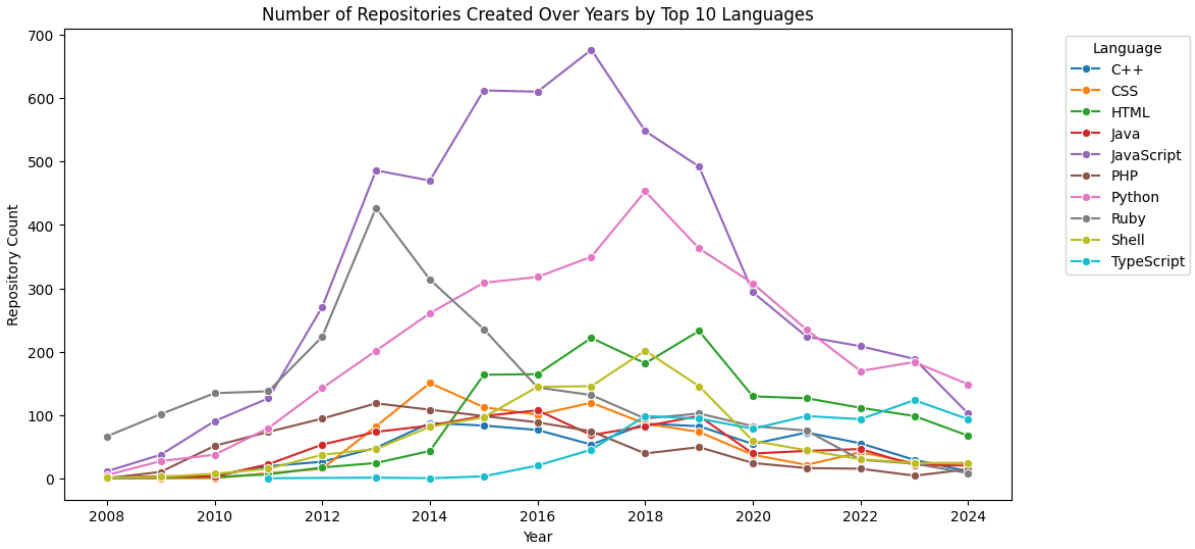
4. Mid-range languages:
    - HTML: ~2,800 repositories
    - CSS: ~1,700 repositories
    - C++: ~1,500 repositories
    - All show moderate adoption of both features

5. Lower adoption languages:
    - PHP, Shell, TypeScript, and Java show lower total counts
    - Still maintain a mix of both Projects and Wikis
    - Generally under 2,000 repositories each

The data suggests that:

- Projects and Wikis are most commonly used in JavaScript and Python ecosystems
- Most languages show higher Wiki adoption than Projects
- Even smaller language communities make use of both features
- Documentation (Wikis) appears to be prioritized over project management features across most languages

13. Language Popularity Over Years (Number of Repositories per Language per Year)



This graph shows the number of repositories created in various programming languages each year from 2008 through to 2024. Here are the key trends:

1. JavaScript dominates in repository creation:
   - Steady growth from 2008-2014
   - Sharp increase from 2014-2016
   - Peak around 2018 (~670 repositories)
   - Gradual decline after 2018
6. Python shows strong growth:
   - Consistent upward trend from 2008-2018
   - Peak around 2018 (~450 repositories)
   - Moderate decline afterward but maintains significant activity
7. Ruby presents an interesting pattern:
   - Started strong in 2008
   - Peaked around 2014 (~420 repositories)
   - Steady decline thereafter
8. HTML shows cyclical patterns:
   - Multiple peaks and valleys
   - Notable peaks around 2016 and 2019 (~220 repositories)
9. Other languages:
   - TypeScript shows gradual growth starting from 2014
   - CSS maintains moderate levels throughout
   - C++, PHP, and Shell show relatively stable but lower numbers

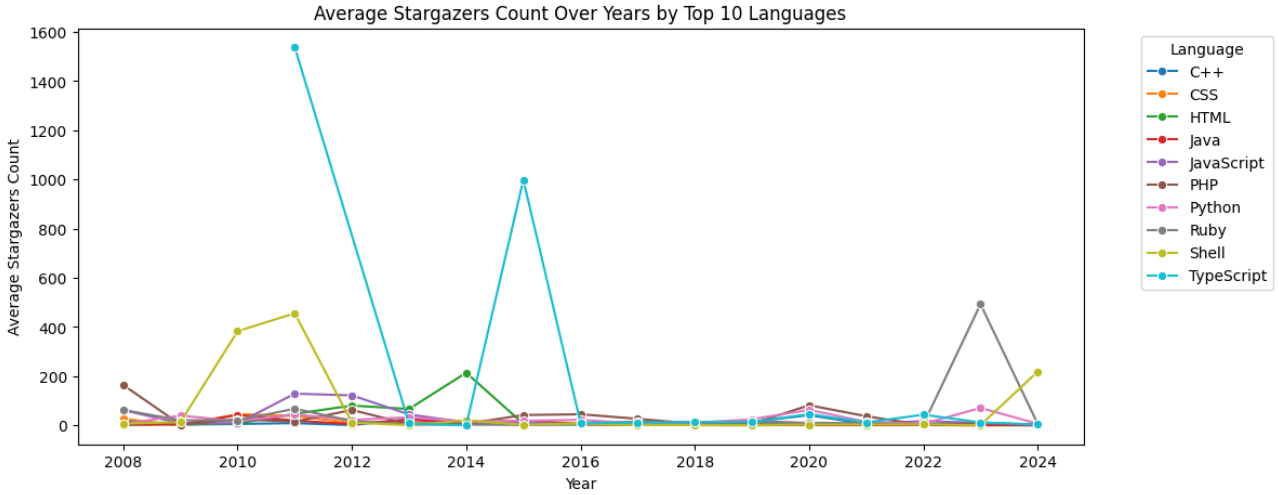
- Java shows moderate activity with slight decline in recent years

The overall trend suggests a shift in the development landscape:

- Most languages reached their peak activity between 2016 and 2019
- General trend of a decline in repository creation for most languages after 2020
- The most recent years have seen a smoothening or evening out in the distribution of most languages.

This could be indicative of either market maturation, the consolidation of projects, or changes in how developers are maintaining their code repositories.

14. Average Stargazers Count Over Years by Top 10 Languages

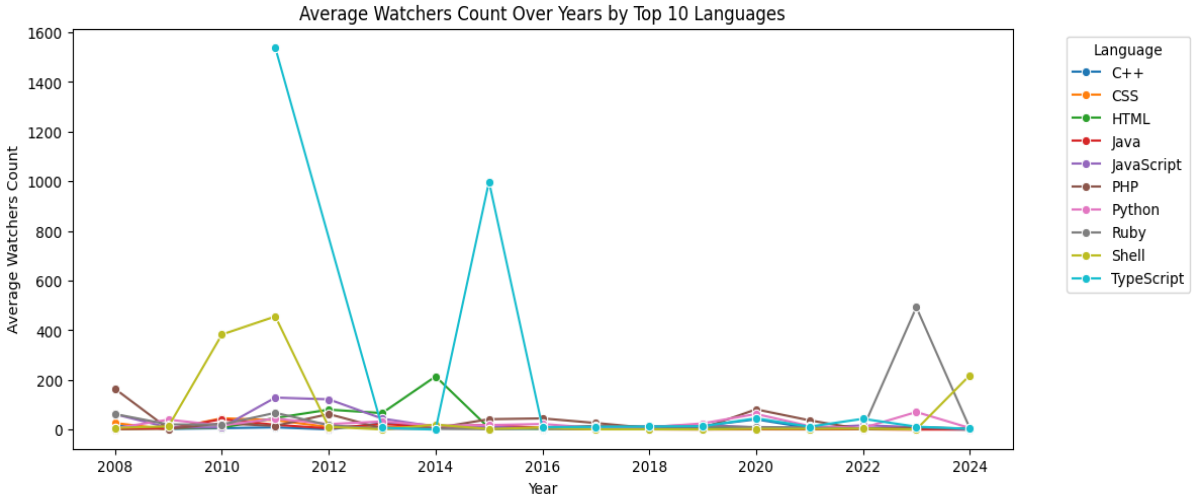


This graph shows the average number of "stargazers" (likely GitHub stars/followers) for different programming languages from 2008 to 2024. Here are the key observations:

2. TypeScript shows the most dramatic fluctuations, with two notable peaks:
   - A major peak around 2012 (~1500 stargazers)
   - Another significant peak in 2016 (~1000 stargazers)
3. Shell scripting had moderate popularity in the early 2010s, with a peak around 2012-2013.
4. Ruby shows an interesting spike around 2022, reaching about 500 stargazers before declining.
5. Most other languages (C++, CSS, HTML, Java, JavaScript, PHP, Python) maintain relatively stable and lower average stargazer counts, typically below 200 throughout the period.
6. Recent trends (2022-2024) show:
   - Shell seeing a slight uptick
   - Python showing a gradual increase
   - Most other languages maintaining relatively stable numbers

The graph suggests that while some languages experience occasional spikes in popularity, most maintain consistent but modest following over time. TypeScript's dramatic peaks might represent major releases or significant developments in the language's ecosystem during those years.

15. Average Watchers Count Over Years by Top 10 Languages



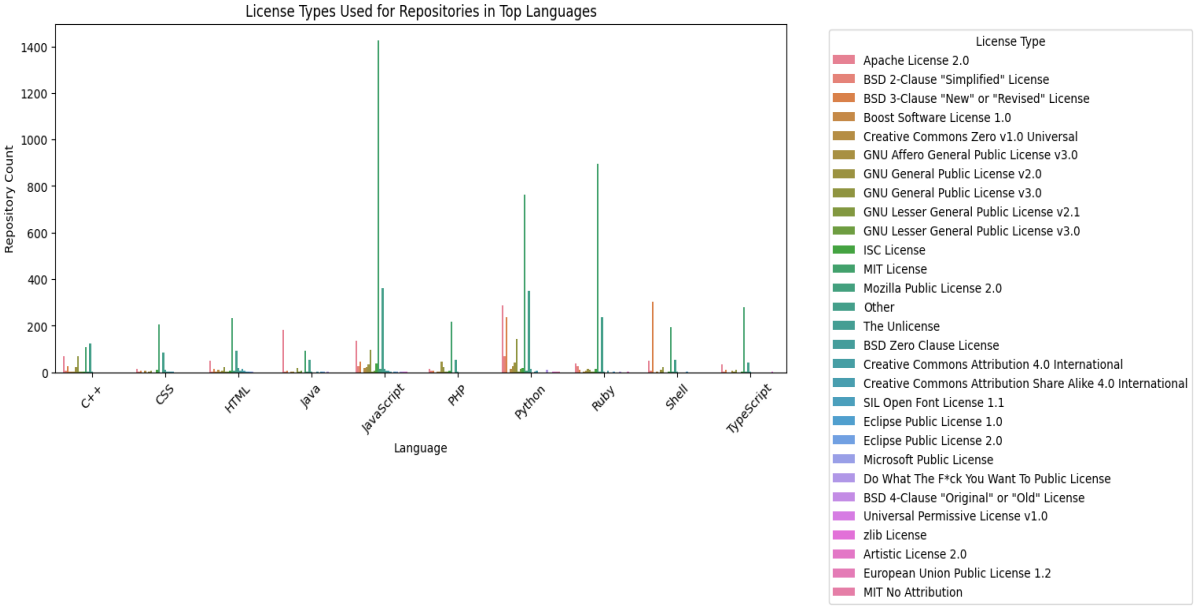
This graph shows the average number of "watchers" for different programming languages over time from 2008 to 2024. On GitHub,

The key patterns are the following:

7. TypeScript shows two major peaks:
   - Highest peak around 2012 (~1500 watchers)
   - Second peak in 2016 (~1000 watchers)
8. Shell scripting had a moderate rise in the early 2010s
9. Ruby shows a distinct spike around 2022 (~500 watchers)
10. Other languages (C++, CSS, HTML, Java, JavaScript, PHP, Python) maintain relatively low and stable watcher counts below 200

The similarity between the watcher graph and the stargazers graph suggests there might be a strong correlation between starring and watching behaviour on these repositories.

16. License Types Used for Repositories in Top Languages Top 10 Languages



This graph shows the distribution of license types across different programming languages in repositories. Here are the key observations:

1. Most Popular Licenses:

   - MIT License appears to be the dominant choice across most languages, showing particularly high peaks for JavaScript (~1400 repositories) and Python (~750 repositories)
   - Apache License 2.0 shows significant usage across several languages
   - GNU GPL variants (v2.0, v3.0) have notable presence

2. Language-Specific Patterns:

   - JavaScript has the highest single concentration of MIT Licensed repositories
   - Python shows the second-highest concentration of MIT Licensed projects
   - Ruby shows a substantial but lower peak of MIT Licensed repositories
   - -TypeScript repositories tend to favor MIT and Apache licenses

3. Less Common Licenses:

   - Creative Commons licenses appear occasionally

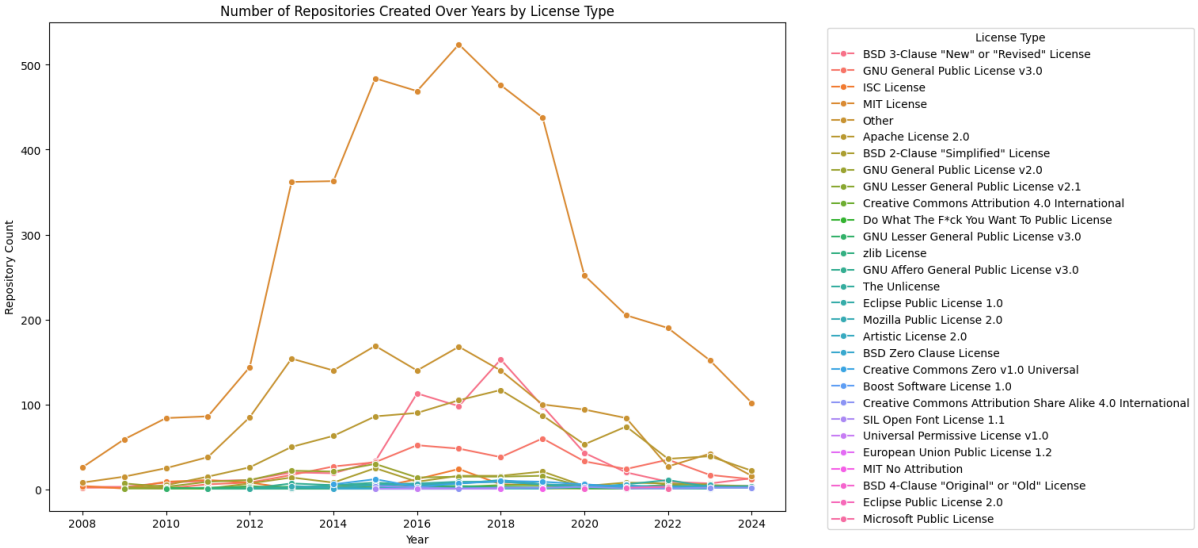
- BSD variants (2-Clause, 3-Clause, Zero Clause) have modest representation
- Specialized licenses like Eclipse, Mozilla Public License, and LGPL show minimal usage

4. Interesting Notes:

- The presence of the very permissive licensing
- Some languages show more diverse license usage than others
- Many specialized licenses (like EU Public License, Artistic License) see very limited adoption

The overwhelming preference for MIT licensing across languages suggests developers generally favor simple, permissive licenses for their open-source projects.

17. Number of Repositories created over years by license type



This graph shows the evolution of license usage in repositories over time from 2008 to 2024. Here are the key trends:

1. MIT License dominance:

- Strong growth from 2012-2016
- Peak around 2017 (~525 repositories)
- Sharp decline after 2018
- Still maintains leadership position despite decline

2. ISC License shows interesting pattern:

    - Significant growth 2012-2014
    - Sustained high levels through 2016
    - Steady decline after 2018

3. Other major licenses:

    - Apache License 2.0 shows moderate but consistent usage
    - GNU GPL v3.0 maintains steady presence throughout
    - BSD licenses (2-Clause and 3-Clause) show stable but lower usage

4. Timeline trends:

    - 2012-2017: Period of strong growth in repository creation across licenses
    - 2017-2018: Peak period for most license types
    - 2018-2024: General decline across most license types

5. Less common licenses:

    - Most specialized licenses (Mozilla, Eclipse, Creative Commons, etc.) maintain very low usage throughout
    - Little variation in their adoption rates over time

The general trend is consolidation around a few major licenses, especially MIT during peak years, and generally a decline in new repository creation across all license types in recent years. This may indicate either market maturation or changes in how developers approach licensing their projects.

4.Conclusion

The overall conclusion about the GitHub ecosystem and programming language trends in Chicago is as follows:

1. Platform Advantage:
    - Users affiliated with GitHub and ShorebirdTech show strongest community reach, likely due to platform synergy.
2. Corporate Impact:
    - Google-affiliated user demonstrates highest repository activity despite newer account, showing strong open-source commitment.

3. Engagement Patterns:
    - Three distinct user types emerge: community builders (high follower/following), content creators (high repos), and influential observers (high followers, low following)
    - Account age doesn't directly correlate with influence or activity levels
4. Activity Sustainability:
    - Newer accounts (particularly Google) show higher activity rates
    - Older accounts maintain consistent but lower activity rates
    - Platform-affiliated accounts maintain balanced engagement metrics
5. Language Popularity and Evolution:
    - JavaScript became dominant in creating repositories and adopting features.
    - Python had a very strong, consistent run-up and continues to hold a strong second place.
    - Traditional languages like Ruby started off strongly but faded over time.
    - Typescript had dramatic spikes in interest, but more moderate sustained usage.
6. Repository Management Trends:
    - Peak activity for repository creation occurred between 2016-2019
    - General decline in new repository creation after 2020 across most languages
    - Higher adoption of Wikis compared to Projects suggests strong emphasis on documentation
    - Larger ecosystems (JavaScript, Python) tend to make better use of GitHub's collaborative features

7. Licensing Patterns:
    - MIT License emerged as the clear favorite across all major languages
    - Permissive licenses generally preferred over restrictive ones
    - Licensing choices peaked around 2017-2018 and then declined
    - Specialized licenses remain niche with very low adoption

8. Community Engagement:
    - Stargazers and watchers show similar patterns, suggesting coordinated community interest
    - JavaScript and Python communities show the highest engagement levels
    - Newer technologies (like TypeScript) can generate intense but temporary spikes of interest
    - Documentation (wikis) is prioritized across all language communities

9. 5. Overall Ecosystem Maturation:
    - The GitHub ecosystem shows signs of maturation with:

- Consolidation around fewer, more established languages
- Standardization of licensing (primarily MIT)
- More balanced distribution of new repositories in recent years
- Decline in the rate of new repository creation
- Strong emphasis on documentation and community collaboration

These trends suggest that an open source ecosystem is maturing-one that favors simplicity in licensing, solid documentation, and established platforms, even while it remains open to new technologies and approaches.