EAWOC: Exploratory Analysis of Commit Messages in World Of Code

Addi Malviya Thakur, Mahmoud Jahanshahi, Rekha Bhupatiraju, Minoo Oliaee, Prachi Patel

I. OBJECTIVE

World of code (WoC) has enabled research on the global properties of Free and open-source software (FOSS)[5]. The objective of this project would be to conduct an exploratory analysis on the ∼1.5 Billion commit messages present in WoC. The exploratory analysis will investigate the following research questions:

RQ-1 Performing *descriptive statistics* to understand the basic features of the commit messages in WoC[4]. In particular, we will gather fundamental insights of WoC by calculating the *measures of central tendencies* and the *dispersion* of the commit messages in the WoC.

RQ-2 *Term frequency extraction* and analysis to discover the most common words used in the commit messages

RQ-3 Plotting the *temporal distribution* of commits in random 1,000 projects with at least 100,000 commits. Also, performing *distribution fitting* to find the best distribution that fits the underlying commit times at each repository level.

RQ-4 Hypothesis testing for the presence or absence of *Pareto Principle* for the number of commits at the ecosystem level. Specifically, we will study if 80% of commits come from 20% of projects.

RQ-5 Develop *static and interactive visualizations* to demonstrate the findings.

II. MOTIVATION

Since its inception, Free/Libre and open-source software (FLOSS) has dramatically impacted the software community and ecosystem. FLOSS is both free software and open-source software where anyone is freely licensed to use, copy, study, and change the software in any way. The source code is openly shared so that people are encouraged to voluntarily improve the design of the software[7]. GitHub is a popular place to host open-source projects and is the largest source code repository globally. It offers a socio-collaborative working environment for open-source projects, where individuals can develop and release software. Adding comments during pull and push updates to the software code is a popular way to discuss the changes and interact with the community. Analyses inform other developers of bug reports, planned improvements, modifications, and feedback, among other activities. WoC allows easy and programmatic access to such comments for research and analysis[5].

An exploratory analysis of comments would allow insight into the development and progress of software, its updates, and current interest and usage. Furthermore, such an analysis would usher novel research methodologies to understand team dynamics, software evolution cycle, identification of common pitfalls - what works and what doesn't, eventually providing a holistic approach towards improving software development and engineering processes.

III. DATASET

The WoC collection of software repository contains cross-reference authors, projects, commits, blobs, dependencies, and history of the FLOSS ecosystems. This data is updated on a regular basis and contains billions of git objects. For the purpose of this study we will use WoC's large and frequently updated collection of version control commit data of the entire FLOSS ecosystems.

WoC API's will be used to fetch and process the data related to comments. We plan to extract comments from repositories with at least 100,000 commits made in a period of time (say, last 5 years). We will also be collecting meta data related to these commit messages including:

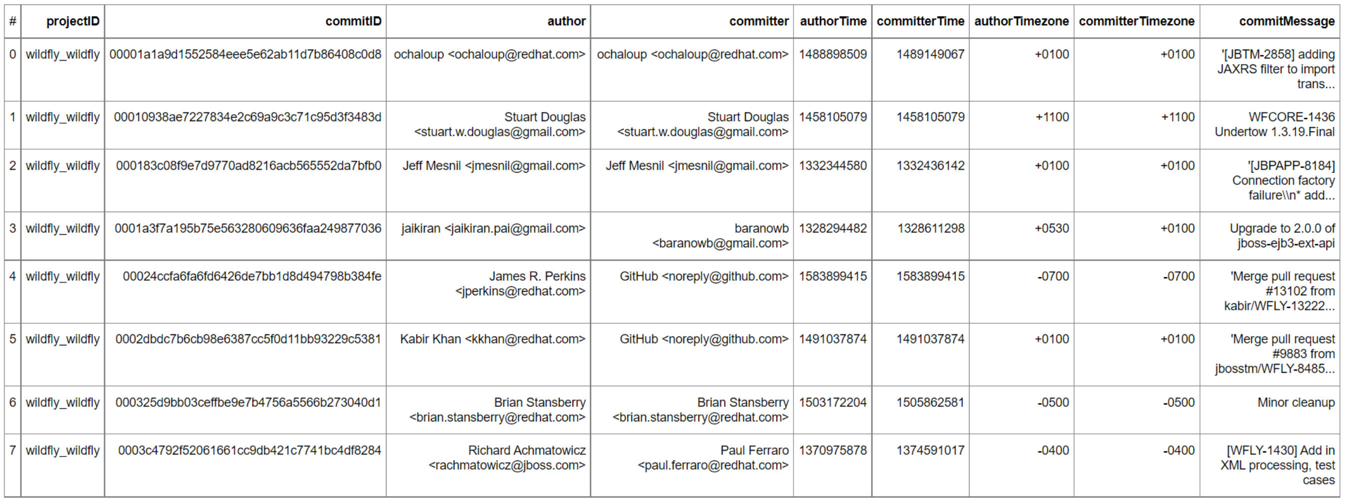
1) commit info: timestamps, number of associated blobs, number of associated files
2) user info: number of projects, number of commits
3) repository info: number of authors, blobs, time since creation

The exact extent to which we plan to dive deep into each of these three areas will depend on the results obtained at each stage of our project.

A. Data collection

We started our data collection by using WoC MongoDB collections. Explicitly, we used projects table version T to find the projects with more than 100,000 commits. It should be mentioned that there are numerous repositories almost entirely based on some parent repository from which they were cloned and could be considered as the same project. WoC projects' data collection uses deforked version of projects [6] which is based on identified clones and aggregated form of the data from these clones in a single project. This filtering resulted in 1156 deforked projects with more than 100k commits which we used as the base for our analysis.

In the next phase, using WoC mappings, we augmented our selected projects with their total number of commits, blobs, original blobs, files, authors, forks, number of months they have been active (the time between their first and last commit) and their last commit time. Having these data, we then retrieved all the commits in these projects. This resulted in total of 488 million commits. We then retrieved the content of each of these commits including their message, author and time.



#	projectID	commitID	author	committer	authorTime	committerTime	authorTimezone	committerTimezone	commitMessage
0	wildfly_wildfly	00001a1a9d1552584eee5e62ab11d7b86408c0d8	ochaloup <ochaloup@redhat.com>	ochaloup <ochaloup@redhat.com>	1488898509	1489149067	+0100	+0100	'[JBTM-2858] adding JAXRS filter to import trans...
1	wildfly_wildfly	00010938ae7227834e2c69a9c3c71c95d3f3483d	Stuart Douglas <stuart.w.douglas@gmail.com>	Stuart Douglas <stuart.w.douglas@gmail.com>	1458105079	1458105079	+1100	+1100	WFCORE-1436 Undertow 1.3.19.Final
2	wildfly_wildfly	000183c08f9e7d9770ad8216acb565552da7bfb0	Jeff Mesnil <jmesnil@gmail.com>	Jeff Mesnil <jmesnil@gmail.com>	1332344580	1332436142	+0100	+0100	'[JBPAPP-8184] Connection factory failure\\n* add...
3	wildfly_wildfly	0001a3f7a195b75e563280609636faa249877036	jaikiran <jaikiran.pai@gmail.com>	baranowb <baranowb@gmail.com>	1328294482	1328611298	+0530	+0100	Upgrade to 2.0.0 of jboss-ejb3-ext-api
4	wildfly_wildfly	00024ccfa6fa6fd6426de7bb1d8d494798b384fe	James R. Perkins <jperkins@redhat.com>	GitHub <noreply@github.com>	1583899415	1583899415	-0700	-0700	'Merge pull request #13102 from kabir/WFLY-13222...
5	wildfly_wildfly	0002dbdc7b6cb98e6387cc5f0d11bb93229c5381	Kabir Khan <kkhan@redhat.com>	GitHub <noreply@github.com>	1491037874	1491037874	+0100	+0100	'Merge pull request #9883 from jbosstm/WFLY-8485...
6	wildfly_wildfly	000325d9bb03ceffbe9e7b4756a5566b273040d1	Brian Stansberry <brian.stansberry@redhat.com>	Brian Stansberry <brian.stansberry@redhat.com>	1503172204	1505862581	-0500	-0500	Minor cleanup
7	wildfly_wildfly	0003c4792f52061661cc9db421c7741bc4df8284	Richard Achmatowicz <rachmatowicz@jboss.com>	Paul Ferraro <paul.ferraro@redhat.com>	1370975878	1374591017	-0400	-0400	[WFLY-1430] Add in XML processing, test cases

Fig. 1. Data Example

B. *Data pre-processing and preparation*

The data extraction process results in 128 CSV files containing several thousands of projects and millions of commits. We converted the input files data in to parquet file format. Apache Parquet is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language[8]. We converted these files to improve the reading and writing of this very large amounts of data that could not fit in the memory. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk.

C. *Integration and Validation*

We found two files that were incompatible to be converted to the parquet extension. The files are then used for out-of-memory loader and processing in python. The data conversion happened using the UTF-8 encoding which has preserved the integrity and overall data quality. Next, we used VAEX data structure as the default data unit for the processing of the data. Vaex is a python library for lazy Out-of-Core DataFrames (similar to Pandas), to visualize and explore big tabular datasets[2]. It can calculate statistics such as mean, sum, count, standard deviation etc, on an N-dimensional grid up to a billion objects/rows per second. Visualization is done using histograms, density plots and 3d volume rendering, allowing interactive exploration of big data. Vaex uses memory mapping, a zero memory copy policy, and lazy computations for best performance (no memory wasted). An example of the data table is shown in Figure-1.

IV. Models and Algorithm

A. *RQ1*

In this approach we perform *descriptive statistics* to understand the basic features of the commit messages in WoC[4]. In particular, we will gather fundamental insights of WoC by calculating the *measures of central tendencies* and the

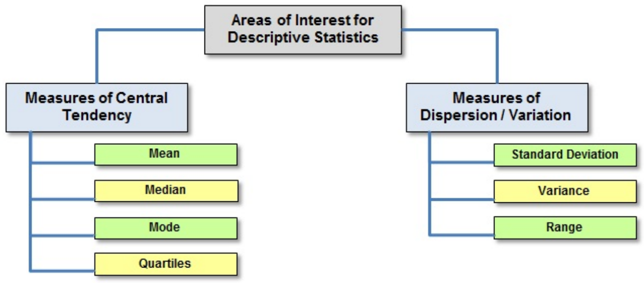

Areas of Interest for Descriptive Statistics
Measures of Central Tendency
Mean
Median
Mode
Quartiles
Measures of Dispersion / Variation
Standard Deviation
Variance
Range

Fig. 2. Descriptive Statistics

dispersion of the commit messages in the WoC. Descriptive statistics are used to describe the basic features of the data in a study. They provide simple summaries about the sample and the measures. Together with simple graphics analysis, they form the basis of virtually every quantitative analysis of data. Measures of central tendency are the most basic and, often, the most informative description of a population's characteristics. They describe the average member of the population of interest. There are three measures of central tendency Figure-2: 1) Mean:- the sum of a variable's values divided by the total number of values; 2) Median:- the middle value of a variable; 3) Mode -the value that occurs most often. Dispersion in statistics is a way of describing how spread out a set of data is. When a data set has a large value, the values in the set are widely scattered; when it is small the items in the set are tightly clustered. Common examples of measures of statistical dispersion are the variance, standard deviation, and inter-quartile range (see table I).

B. *RQ2*

This research questions investigates the relevant words of a commit in a collection of commits. This provides insights in to the construction of commit messages, their relevance to understanding the commit, and evaluate how important a word is to a commit in a collection or corpus of commits.

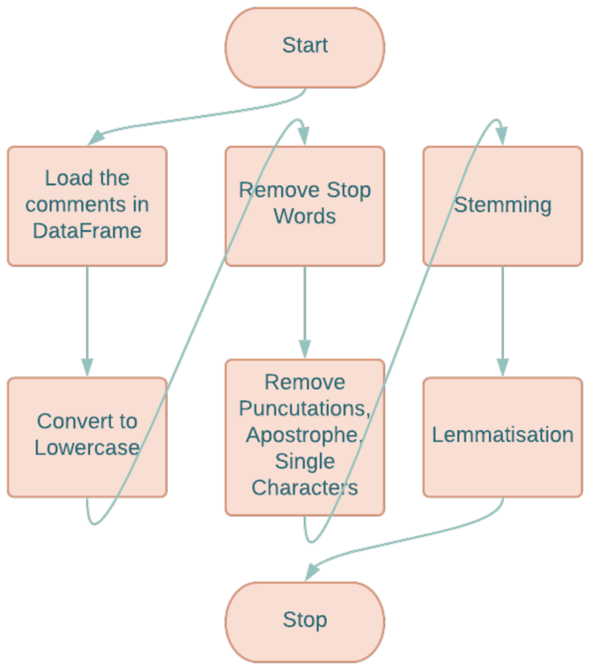

Start
Load the comments in DataFrame
Remove Stop Words
Stemming
Convert to Lowercase
Remove Puncutations, Apostrophe, Single Characters
Lemmatisation
Stop

Fig. 3. Workflow for data preparation for TF-IDF

The importance increases proportionally to the number of times a word appears in the commit but is offset by the frequency of the word in the corpus. Variations of the TF-IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a commits' relevance given a user query. The preliminary workflow to prepare the data for TF-IDF computation is shown in Figure-3.

TF-IDF can be broken down into two parts TF (term frequency) and IDF (inverse document frequency). *TF* works by looking at the frequency of a particular word relative to the commit. The number of times a word appears in a commit divided by the total number of words in the commit. Every commit has its own term frequency.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

IDF is the log of the number of commits divided by the number of commits that contain the word w. Inverse data frequency determines the weight of rare words across all commits in the corpus.

$$idf(w) = log(\frac{N}{df_t})$$

The IDF is computed once for all commits. Lastly, the TF-IDF is simply the TF multiplied by IDF.

$$w_{i,j} = tf_{i,j} \times log(\frac{N}{df_i})$$

In this study, we plan to use the sklearn provided implementation of TF-IDF ($TfidfVectorizer$).

C. RQ3

We use our commit time data to explore the temporal distribution of the commits in each project. We do this by

TABLE I
AGGREGATE STATISTICS

Type	**Statistics**
Number of projects	1161
Minimum number of commits	2
Maximum number of commits	31332118
Average number of commits (Mean)	415500
Median number of commits (Median)	177880
Most common number of commits (Mode)	8
Harmonic mean of commits	253.5
25% IQR	124785
75% IQR	332968
Variance (commits)	1567587572186
Population Variance (p_variance)	1566237367559
Standard deviation (commits)	1252033
Population Standard Deviation (p_stdev)	1251494

randomly selecting 1000 of the repositories and performing a distribution fitting on all the commit times for it. The library used for these fittings performs a fitting using the sum of squares error $\sum_{i=1}^{n}(x_i - \overline{x})^2$ on each distribution from a list of common distributions, and selects the one with the smallest error. [3] For further analysis, we referred back to the raw data using project names.

D. RQ4

In this section, we test the null hypothesis for the presence or absence of *Pareto Principle* for number of commits at the ecosystem level. Specifically, we will study if 80% of commits come from 20% of projects. Initially, we begin by testing the hypothesis of the commits made at the aggregate project level. Then we perform the same hypothesis testing at the author level for a random sample of 50 repositories (at each repository level). The Pareto principle is an illustration of a *power law* relationship, because it is self-similar over a wide range of magnitudes, it produces outcomes completely different from Normal or Gaussian distribution phenomena. To study the hypothesis we will test the power law fitting.

The power laws are the probability distribution with the following form

$$p(x) \propto x^{-\alpha} \tag{1}$$

This heavy tailedness can be so extreme that the standard deviation of the distribution can be undefined (for $\alpha < 3$), or even the mean (for $\alpha < 2$). So, we will test the presence of power law as a value ($2 < \alpha < 3$) for the presence of power law (or Pareto principle)

V. RESULTS AND ANALYSIS

In this section, we address the science questions mentioned in the objectives. First we perform descriptive statistics to understand the basic structure of the dataset, followed by the details study of the text of commits to examine the general patterns that commits are based on. Thirdly, we look in to the temporality of messages as a function of project dynamics or stagnation, followed by studying the project activities and importance through power-law fitting. The visualization objective is spread across all the aforementioned objectives.

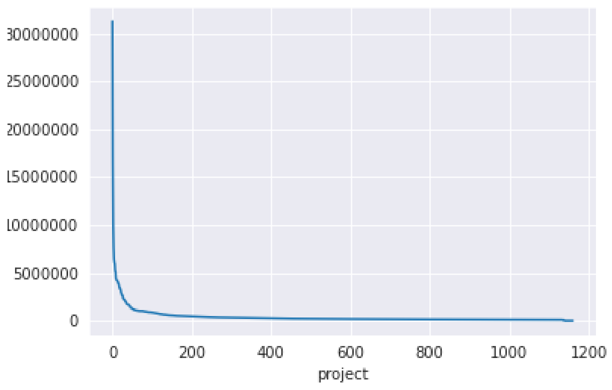


Fig. 4. Commits messages distribution

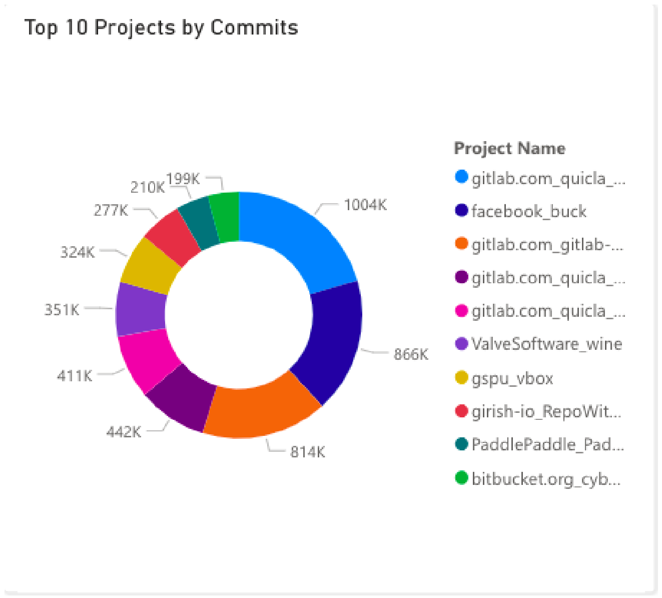


Fig. 5. Top 10 Projects by Commits

A. RQ1

1) Aggregate Statistics: This RQ generates two types of results. First we generate the overall descriptive statistics of the dataset, followed by a statistics of a random sample of 1000 projects. As shown in Table-I, the overall descriptive statistics reveal an average number of commits to a project, and least and the most number ones. The total number of projects studies are 1161, while the mean number of commits around 400K, while the most are 31 million commits. In Figure-4, we have shown the distribution of aggregate number of commits messages for each project. The results indicate a potential heavy-tailed distribution that we will investigate in RQ-4.

RQ1-Visualization: In Figure-5, we showed the top 10 projects by commits and in Figure-7, we showed the distribution of 30 project repositories. We found very projects have a large number of commits, while majority of them few hundred commits. The descriptive statistics for all the project together is shown in Figure-6 .

TABLE II
DISTRIBUTION FITTING.

Distribution	Mean	Median
powerlaw	2657.34	247.0
norm	4831.95	538.0
cauchy	2369.81	296.0
gamma	4069.81	569.0
rayleigh	2276.37	282.0
uniform	340.50	1.0
expon	87.28	10.5
lognorm	45264.83	135.5
xponpow	1283.18	7.0
chi2	1.0	1.0

B. RQ2

We begin by randomly identifying 100 projects and 100 random commit messages for each of the 100 projects. We aggregated and compiled over 100,000 message to generate the TF-IDF of these messages. These message were cleaned and the standard english based stop words were removed from the corpus. Next, we used sklearn *TfidfVectorizer* library to perform the TF-IDF generation of the commits.

RQ2-Visualization: We discover 9548 unique features that are spread across 100,000 commits. The results of the TF-IDF is shown in the wordcloud in Figure-8

C. RQ3

We find that a plurality of commits follow a powerlaw distribution, and that normal, cauchy, gamma, and rayleigh distributions are also common. We can see the number of each in Figure-9. Because there is no overwhelming majority, we explore other features of the data that might be correlate with the type of distribution. It seems normally and gamma distributed projects have more authors than powerlaw, cauchy and rayleigh. The other distributions do not have enough projects to draw any conclusions. Specific mean and median numbers of authors can be seen in figure 10. Finally, the time over which a project is done could be a factor in what distribution it takes. As an initial investigation we just compared the scale of the distributions, but this is an imperfect measure as scale is dependent on the type of distribution to some degree. Still, by visual analysis of Figure-11, we see that powerlaw and gamma distributions seem to occur for longer term projects, while normal, cauchy, and rayleigh distributions occur over shorter terms. This makes sense as the longest term projects would grow in scope over time rather than tapering down as they reach completion.

RQ3-Visualization: The results indicate heavy tailed distribution dominates the projects and related fits the underlying commit distribution. We also investigated the number of authors in projects per distribution type that are shown in Figure-10. Finally, we generated basic statistics of the fitting that are shown in the Table-II.

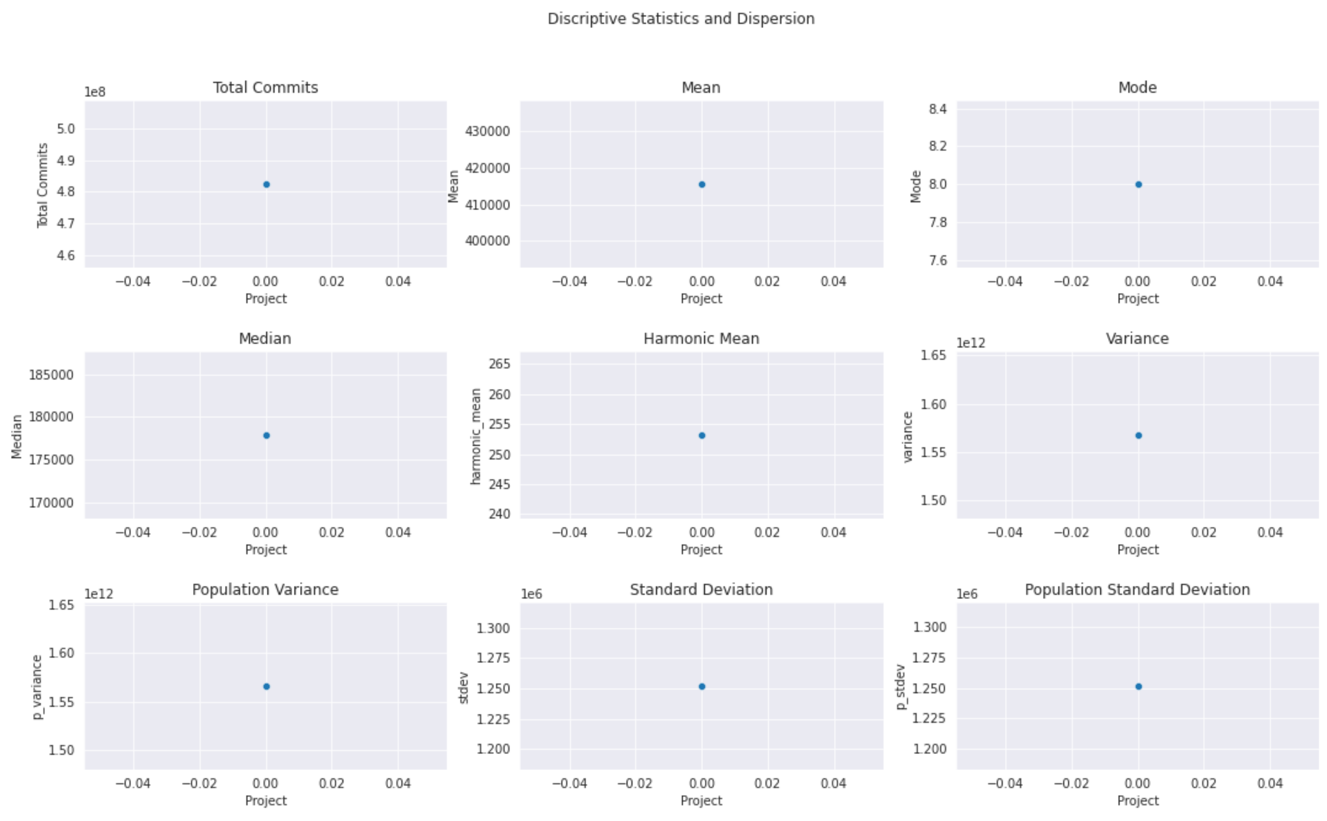


Fig. 6. Statistics for all projects

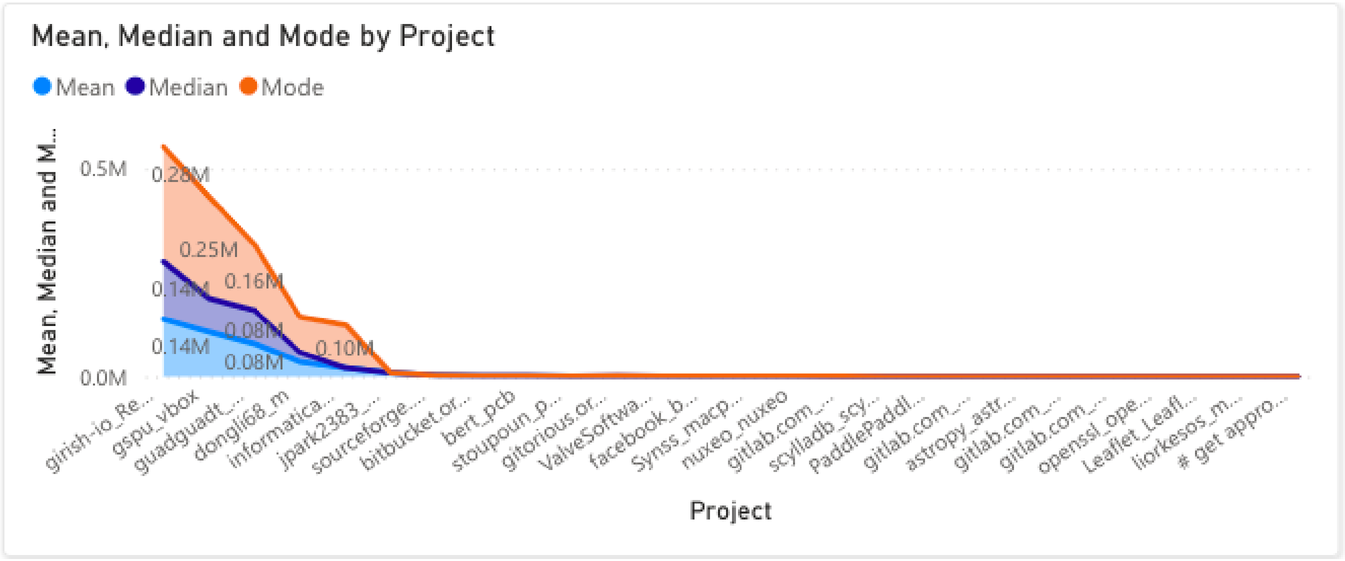


Fig. 7. Mean, Median and Mode of random 30 projects

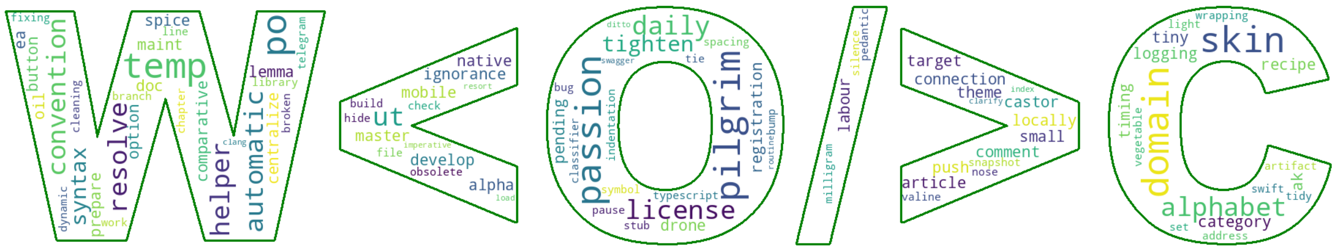


Fig. 8. TF-IDF results of the commit messages.

Distribution	No. of projects
powerlaw	303
norm	206
cauchy	165
gamma	143
rayleigh	101
uniform	30
expon	28
lognorm	12
exponpow	11
chi2	1

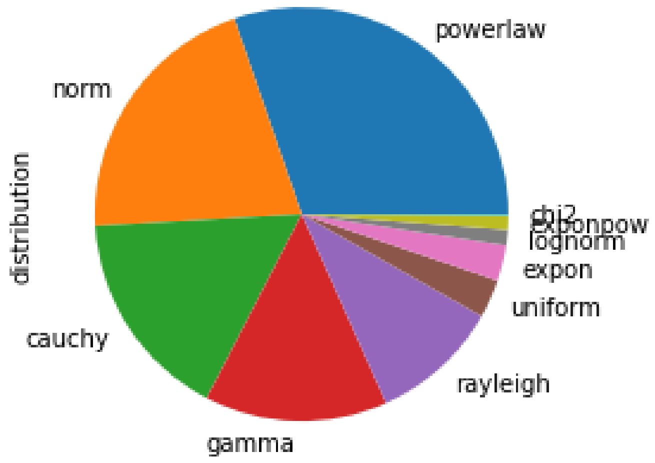


Fig. 9. Number of projects in each distribution out of 1000

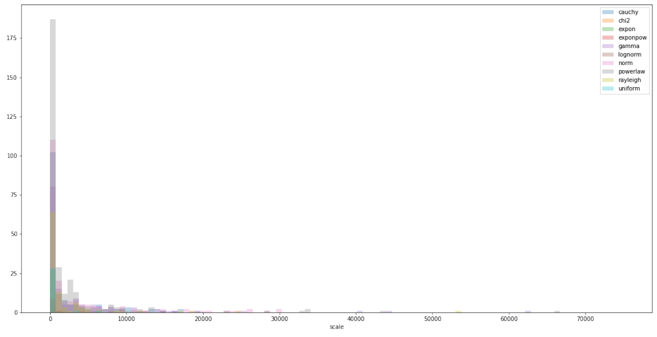


Fig. 10. Number of authors in projects per distribution type

D. RQ4

We begin by modeling the power law distribution of aggregating the commit messages at the project level. Figure-12 shows the distribution of commit messages at the project level. This distribution while plotted on the normal scale does indication potential presence of a heavy-tailed distribution in the commit messages. Next, we used the power law testing approach as described in [1]. The resulting power law fitting plot in shown in Figure-13 with $\alpha = 2.17$ that clearly shows there are certain projects with very high number of commit messages while a majority of the projects have just enough number of commit messages, thereby proving our Pareto principle hypothesis.

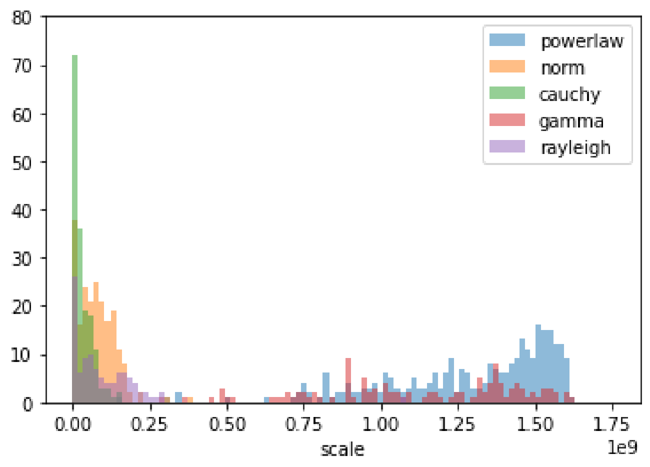


Fig. 11. Scale of each of the common distributions

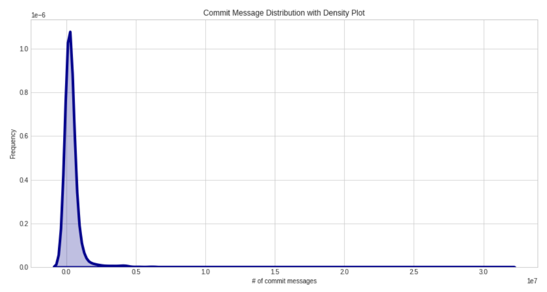


Fig. 12. Commits distribution aggregated at the projects

RQ4-Visualization: The results of modeling the Pareto principle is shown in Figure-13 and 14. These figure shows that the aggregate statistics of commit messages at the projects shows heavy-tailed fitting, while only a few individual projects have their own power law fitting.

VI. Issues Encountered

In this section, we illustrate issues and challenges encountered during the project execution.

1) Data curation: The biggest challenge was extraction and conversion of World of Code data for analysis purposes. Given the size and enormity, it took the team several weeks to work on the data preparation step.
2) Data Analysis: Since the total size of data exceeds the memory capacity, we have to use lazy loading and out-of-core data-frames to investigate the research questions and perform analysis. This was a bit slower and time consuming.
3) Data filtering: The comments and related information associated with the commit messages were vague, unreadable at the same time. It gave a great opportunity assess the execution and creation of relevant tools
4) Assessing the Pareto principal: The decrease on the x-axis and its alignment with Power-law is difficult.

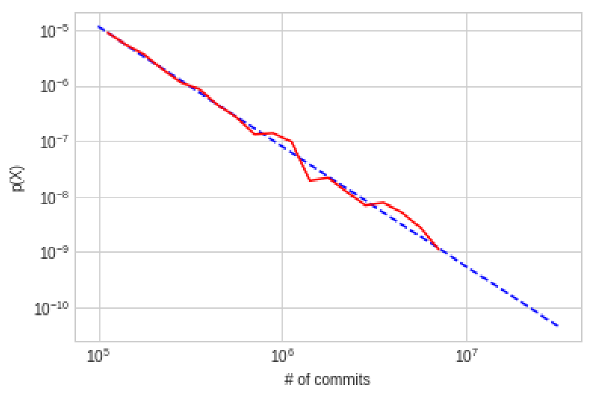


Fig. 13. Power law distribution

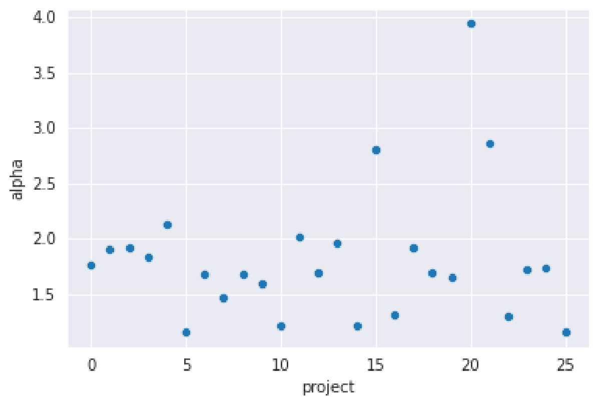


Fig. 14. Power law distribution of α

VII. Conclusion and Future Work

In this work, we studied the Free/Libre and open-source software(FLOSS) that has a tremendous impact on the software community and its ecosystem. In particular, we performed an exploratory analysis of comments to gain insights into the development of software and tool, their updates, and current interest and usage. We extracted aggregated descriptive statistics of thousands of repositories to study general commit patterns at the repository level. We use natural language processing to examine the commit messages, constructs, and word composition. We looked into the evolution of projects through time-series analysis of commits, and finally, we examined the presence of Pareto principles for the commits messages. Our study has highlighted a critical analysis of software development's ecosystem and evolution. In the future, we would like to perform an investigation at the individual repository level and use the finding to ascertain the corresponding project's dynamics.

VIII. TIME-LINE OF MILESTONES

In Table-III, we have shown the timeline for our project leading up the submission of final report during the exam week. The listed task corresponds to the objectives proposed earlier in the project proposal and the timeline is tracked at a weekly scale until the end of November and the first few weeks of December.

We began with the data collection and curation from WoC repository. This process involved extraction, pre-processing, cleaning, and converting in a format for exploratory data analysis. Several of tasking as shown were executed in parallel for efficiency purposes. Efforts that includes visualization and reporting writing were done on a regular basis to include the updates towards final deliverable. The final version of the report was generated on the day of its submission.

IX. ROLES AND RESPONSIBILITIES

We take a team-driven approach to address the objectives of this project. The roles and responsibilities allow for rapid turnout and benefit from individuals' specialized skill-sets and interests. To this end, the following are initial RQs' roles and responsibilities.

- Addi Malviya Thakur: Led the data analysis for the project and assisted in data preparation and visualization. She primarily works on objectives #1, #2, and #4.
- Mahmoud Jahanshahi: Led the data collection for exploratory analysis and served as an interface to access the WoC data through queries and scripting. He also contributed to Objective #1, #3, and #4.
- Rekha Bhupatiraju: Led the #3 and the visualization and plotting of results and findings for the project. She assisted on objectives #1 - #5 and primarily on #5.
- Minoo Oliaee: Contributed to the data preparation for exploratory analysis and the data collection and curation of the WoC data through queries and scripting. She assisted in Objective #1, #2, and #4.
- Prachi Patel: Led the development of visualization dashboard. Also, contributed to the visualization and plotting of results and findings for the project. She assisted on objectives #1 - #5 and closely teamed up on #5.

References

[1] Jeff Alstott, Ed Bullmore, and Dietmar Plenz. powerlaw: A python package for analysis of heavy-tailed distributions. *PLOS ONE*, 9(1):1–11, 01 2014.

[2] Maarten A Breddels and Jovan Veljanoski. Vaex: big data exploration in the era of gaia. *Astronomy & Astrophysics*, 618:A13, 2018.

[3] Thomas Cokelaer. Fitter documentation¶, 2019.

[4] Zealure Holcomb. *Fundamentals of descriptive statistics*. Routledge, 2016.

[5] Yuxing Ma, Tapajit Dey, Chris Bogart, Sadika Amreen, Marat Valiev, Adam Tutko, David Kennard, Russell Zaretzki, and Audris Mockus. World of code: Enabling a research workflow for mining and analyzing the universe of open source vcs data, 2020.

[6] Audris Mockus, Diomidis Spinellis, Zoe Kotti, and Gabriel John Dusing. A complete set of related git repositories identified via community detection approaches based on shared commits. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pages 513–517, 2020.

[7] Walt Scacchi. Free/open source software development. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 459–468, 2007.

[8] Deepak Vohra. Apache parquet. In *Practical Hadoop Ecosystem*, pages 325–335. Springer, 2016.

TABLE III
PROJECT TIMELINE

Task/Timeline	9/27/2021	10/4/2021	10/11/2021	10/18/2021	10/25/2021	11/1/2021	11/8/2021	11/15/2021	11/22/2021	11/29/2021	12/6/2021	Exam Week
Data Collection and Curation	🟩	🟩										
Descriptive Statistics	🟩	🟩	🟩	🟩								
Term Frequency Analysis			🟩	🟩	🟩	🟩						
Temporal Distribution Analysis					🟩	🟩	🟩	🟩				
Distribution Fitting						🟩	🟩	🟩				
Hypothesis Testing for Pareto Principle								🟩	🟩			
Interactive Visualizations		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Report Writing				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

APPENDIX

```python
import pandas as pd
import chardet
import nltk
from nltk.corpus import stopwords
import pickle
import pyarrow.parquet as pq
import numpy
import os
import vaex as vx
import matplotlib.pyplot as plt
import seaborn as sns
import statistics as st
import math


# COLUMN NAMES
columns = ["projectID", "commitID", "author", "
    committer",
           "authorTime", "committerTime", "
    authorTimezone",
           "committerTimezone", "commitMessage"]


# BLOCK FOR READING THE CSV FILE AND CONVERTING
    THEM IN TO PARQUET FILES
import glob

txtFileLocation='/data/fdac21/eawoc//processed_data
    /'

txtfiles = []
for file in glob.glob(txtFileLocation + "/*.txt"):
    txtfiles.append(file)

print('# txt file found:' , len(txtfiles) , '\n',
    txtfiles)
#dd = pd.read_csv('../processed_data/commits.34.txt
    ', sep=';', encoding="ISO-8859-1", low_memory=
    False)


# NOW CONVERT
for file in txtfiles:
    print('Now processing:' + file)
    # SAVE THE CSV IN PARQUET FORMAT
    df_data = vx.from_csv(file, sep=';',
                  encoding="ISO-8859-1",
                  on_bad_lines='skip',
                  names=columns,
                  header=None,
                  dtype='unicode',
                  low_memory=False)
    df_data.export_parquet(file + ".parquet")

print('Conversion done')
```

Listing 1. Parquet Format Conversion