Rupika Dikkala
CS 560: Data Driven SE

MSR2 Writeup

Part 1

Methods

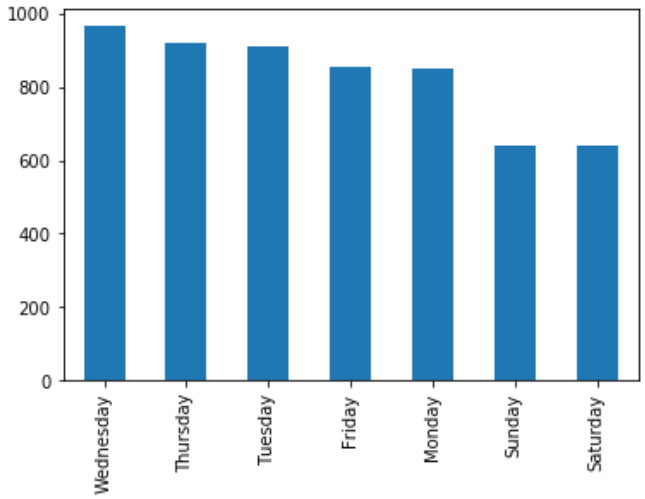
After installing required dependencies and importing the libraries, I authenticated the Github API in order to have a higher-rate limit for submitting requests to the API. Due to the size of the data I was extracting from the bitcoin/bitcoin repository for this assignment, I wrote helper functions to read the fetched issue JSON data into a local file. This helped with run-time efficiency and validity of the data. I extracted all the issues for bitcoin/bitcoin by using a GET command for "/repos/bitcoin/bitcoin/issues" to fetch all the issues in the repository. In order to filter out the pull requests, I dropped all the rows where there was a "null" for the "pull_request" key. After doing so, I dropped the "pull_request" column all together to simplify the dataset.

Q1

Methods

In order to identify which day of the week received the most issues, I took the "created_at" column as that depicts the timestamp a new issue was *created* and converted the timestamp string into a datetime object. Then, I added a new column that fetched the day of the week the issue was created. From here, I took the value_counts() on the 'day' key (which output a frequency table) and plotted it.

Results



Report: *x-axis* represents the days of the week and the *y-axis* is the number of issues. Wednesdays receive the most number of new issues, as the bar graph represents.

Q2

Methods

Number of Comments

In order to get the descriptive statistics for the number of comments per issue, I ran the "describe", "median", "var", and "mode" functions on the "comment_count" column (that I renamed) in the issues dataset.

Word counts of the Issue text

I wrote a function "get_word_count" that first stripped the issue_body text of all the carriage returns and punctuation symbols (periods, commas, etc). From there, I removed the stop-words from the strings, and returned the length of the filtered string. I also made sure to pass the "issue_body" string in all lowercase to avoid case sensitivity issues. The length from the function is then added to the new 'issue_wc' (issue word count) column in the table. I got the descriptive statistics in the same way as "Number of Comments".

Word counts of Comments

I iterated through each issue, and fetched the comments. From there, I passed the comment text into the get_word_count function in order to get the word count without stop words, and appended it to a new "comment_wc" column in the dataframe. This process was rather tumultuous as the script had to iterate through ~5900 issues; it exhausted the API rate limit. Before running descriptive statistics (ran similar to Number of Comments

section), I removed all empty rows in the “comment_wc” column to mitigate calculation issues.

Results

Number of Comments

Metric	Value
Mean	5.29
Median	3.00
Mode	1.99
Standard Deviation	6.87
Variance	47.20

Report: The number of comments per issue is relatively low ($M = 5.29$, $SD = 6.87$).

Word counts of the Issue text

Metric	Value
Mean	115.49
Median	52.00
Mode	20.00
Standard Deviation	378.26
Variance	143083.30

Report: The word count of the issue body text is relatively high ($M = 115.49$, $SD = 378.36$). The variance is extremely high, indicating a far outlier.

Word counts of Comments

Metric	Value
Mean	207.91
Median	66.00
Mode	2.00
Standard Deviation	802.92
Variance	644672.96

Report: The word count of the issue comment text is relatively high (M = 207.91, SD = 802.92).

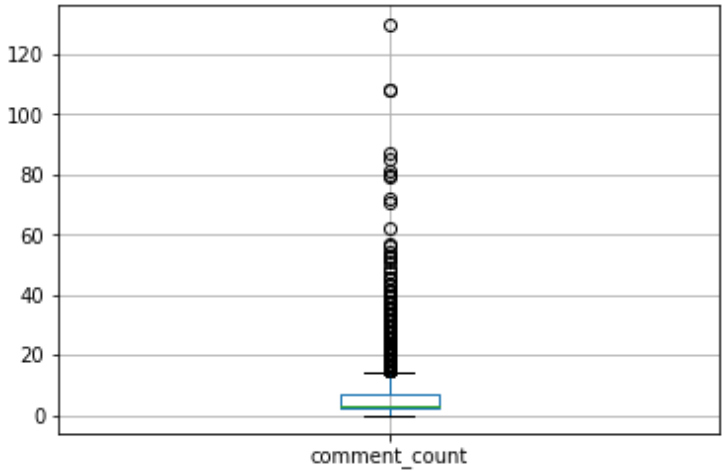
Q3

Methods

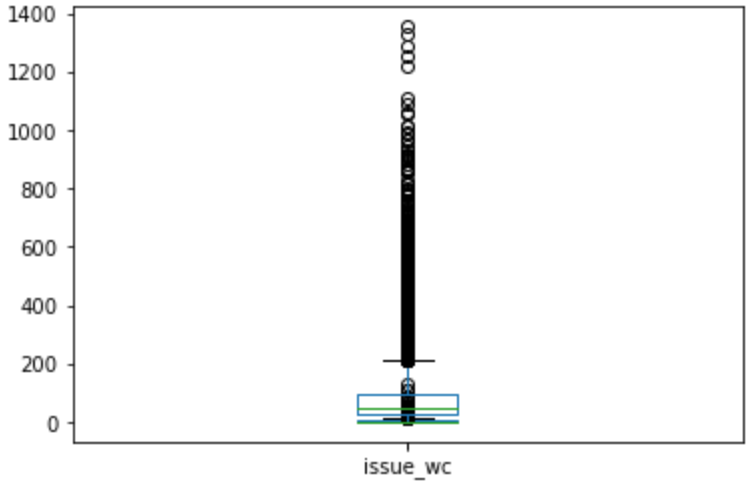
Before I selected a boxplot to represent each metric mentioned in Q2, I made sure to remove any outliers outside of a standard deviation of 3.29 in order to remove variability from the data.

Results

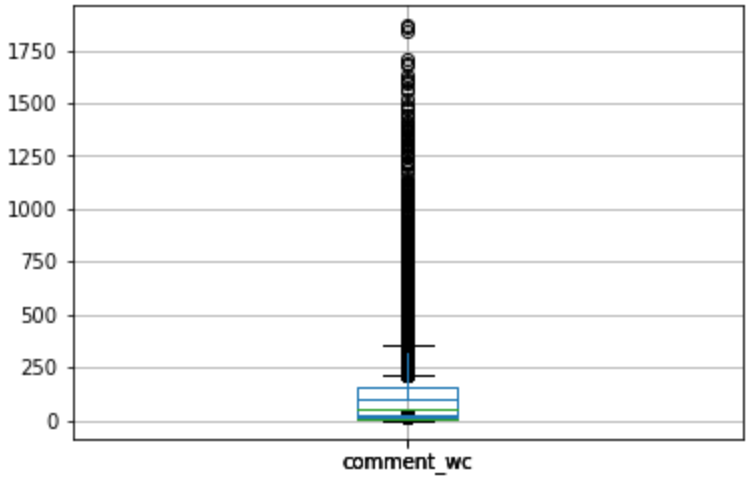
Number of Comments



Word counts of the Issue text



Word counts of Comments



Q4

Methods

After copying over essential columns into a dataframe, I dropped all columns that had an open issue. Once I made minor manipulations, I converted the string timestamps into datetime objects in order to get the duration between opening and closing an issue. The duration is recorded in hours. From there, I ran a Shapiro-Wilk Normality Test to check for normality. After seeing that $p < 0.05$, I decided to go with the Spearman's Rank Correlation Coefficient nonparametric test as the level of measurement was not continuous for either word-count variable (both are ordinal values). I checked for linearity by performing the "straight-line" test on the scatter plot, but that failed as shown in the scatter plot below. Furthermore, the Related Pairs assumptions also fails, because there is no guarantee that the word count will be unique for all issue comments. These points confirm that the Spearman's Rank was the correct choice.

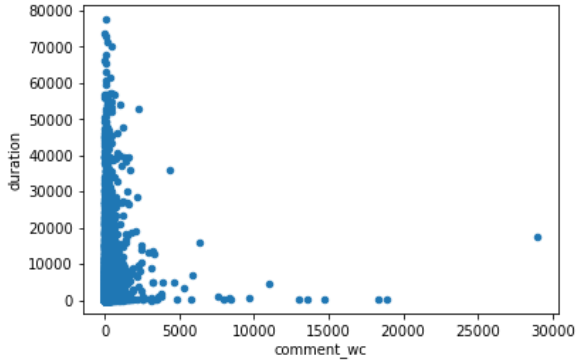


Figure. Scatter plot

Results

Word counts of the Issue text

Data were tested for normality, and yielded $p = 0.0$ making this statistically significant. Using the Spearman's Rank Correlation Coefficient, we found that there is little to no correlation between the word count in the body of issue text and the duration the issue remained open (correlation coefficient = 0.062, p = 8.25e-06)

Word counts of Comments

Data were tested for normality, and yielded $p = 0.0$ making this statistically significant. Using the Spearman's Rank Correlation Coefficient, we found that there is a moderate to medium correlation between the word count of the comment text and the duration the issue remained open (correlation coefficient = 0.321, p = 4.58e-125)

Threats to Validity

I believe my get_stop_words() function is a threat to validity because of the way I mined the words from the repository. I believe I overcompensated when trying to strip off stray punctuation from the strings because there are instances where special characters inform us more about context. For example, a url (https://www.google.com) or an issue number (ex: #1234) should remain as one “word” and shouldn’t be separated. This is an oversight in this assignment, and is (presumably) the reason behind the extremely large variances.