**Section 3: Results**

Please include all required outputs (figures, tables, and/or numbers) in this section.

**Section 4: User Manual/Guide**

**Description**

This program contains scraper, database, and visualization modules meant to assist the user with various tasks. The tasks of each included module are:

1. Scraper: retrieving paper title, author list, publication time, and abstract from PUBMED for a given keyword within a pre-specified time window and saving the retrieved data in the CSV format.
2. Database: importing the CSV file to SQLite to build a database automatically and implementing SQL code to query publications by author name.
3. Visualization: reading the CSV file, showing the number of publications in each month, generating and visualizing the summary statistics for the publication numbers per month, and visualizing the trend of the publication numbers over time.

**Prerequisites**

1. Load up the prerequisites into your Python environment. The code shown below is meant to be added into a Python file or run in a Jupyter Notebook cell. **pm\_query** is the program query where the necessary module functions are stored. This must be imported to use any module functions.

**import** **yaml**

**import** **json**

**import** **os**

**import** **sqlalchemy** **as** **sql**

**import** **pandas** **as** **pd**

**import** **plotly.express** **as** **px**

**import** **pm\_query** **as** **pq**

**from** **Bio** **import** Entrez

1. Install any other missing prerequisite modules or packages.
2. Create a yaml file that contains your NCBI personal API keys and passwords. This is necessary to maximize the efficiency of the scraper module. Use of the yaml file also allows the user to avoid having to hardcode personal data into the Python script.

**Usage**

**Scraper Module**

1. Set the initial parameters of the scraper module. Using function **secret\_manager**, read your yaml file containing your NCBI API keys and passwords. Set the Entrez email parameter and assign your desired search term. For the purpose of the user manual, we will assign the hypothetical search term as “example”.

keys = pq.secret\_manager("api\_keys\_file.yaml")

email = "your\_email@email.com"

search = "example"

1. Gather the data for the final dataframe. Use the **get\_pmid** function to query the eSearch endpoint of the Entrez API to retrieve the corresponding pmids and join them to the input dataframe. Input your desired time window as mindate (start date) and maxdate (end date) using the yyyy/mm/dd format.

example\_pmids = pq.get\_pmid(contact=email, key=keys["apikeys"]["ncbikey"]["key"], term=search, mindate="yyyy/mm/dd", maxdate="yyyy/mm/dd")

1. Using the retrieved pmids, call the **get\_data** function to query the eFetch endpoint to retrieve the details for the corresponding citation as a list of dictionaries.

example\_records = pq.get\_data(pmid\_list=example\_pmids, contact=email, key=keys["apikeys"]["ncbikey"]["key"])

1. Convert the data from the python dictionary into a JSON-encoded object and save the file.

**with** open('example\_records.json', 'w') **as** outfile:

json.dump(example\_records, outfile)

1. Open the saved JSON object from part 4 and parse into a dictionary.

**with** open(‘file\_path\example\_records.json', 'r') **as** outfile:

example\_records = json.load(outfile)

1. Clean the retrieved data by executing the **clean\_data** and **keep\_cleaning** functions. The **keep\_cleaning** function performs additional cleaning on the data by resetting the index of the dataframe, converting the pmid variable to an integer data type and formatting the dates into the %Y-%m-%d’ format. The columns for title and abstract are then joined by index.

search\_clean = pq.clean\_data(example\_records)

search\_clean = pq.keep\_cleaning(example\_clean)

1. Convert the information from the cleaned dataframe into CSV format using the **file\_downloader** function.

pq.file\_downloader("example\_records\_clean.csv", example\_clean)

**Database Module**

1. Use the **csv\_bnb** function. The function reads the CSV file created by the data crawler via the pandas **read\_csv** function. This data is then reformatted for use with SQLite and saved as a new CSV file (called example\_csv).

example\_csv = pq.csv\_bnb("example\_records\_clean.csv")

1. Use the **sqlite\_out** function. The function will take the example\_csv file and use the **create\_engine** function included in sqlalchemy to automatically build a database from the aforementioned file, specifying SQLite as the database dialect.

pq.sqlite\_out(example\_csv)

1. To restrict results to those with a similar author name, call on the **sql\_author\_query** function.

sql\_df = pq.sql\_author\_query("Name")

1. Use the **head** function to display the first 10 results from the query.

sql\_df.head()

**Visualization Module**

1. To either display number of publications in each month as a bar graph, visualize the trend of the publications over time as a line graph, or view both simultaneously as the line graph overlays the bar graph, call on the **draw\_graph** function.

pq.draw\_graph(example\_csv)

1. Use the **summary\_stats** function to create and display the summary statistics by month. Input your desired month. The example month used below is January.

summary\_stats = pq.summary\_stats(example\_csv, "january")

summary\_stats